To appear in Proc. 2013 IEEE Intl. Conf. on Robotics and Automation.

Anytime Solution Optimization for Sampling-Based Motion Planning

Ryan Luna, Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki

***Abstract*— Recent work in sampling-based motion planning has yielded several different approaches for computing good quality paths in high degree of freedom systems: path shortcutting methods that attempt to shorten a single solution path by connecting non-consecutive configurations, a path hybridization technique that combines portions of two or more solutions to form a shorter path, and asymptotically optimal algorithms that converge to the shortest path over time. This paper presents an extensible meta-algorithm that incorporates a traditional sampling-based planning algorithm with offline path shortening techniques to form an anytime algorithm which exhibits competitive solution lengths to the best known methods and optimizers. A series of experiments involving rigid motion and complex manipulation are performed as well as a comparison with asymptotically optimal methods which show the efficacy of the proposed scheme, particularly in high-dimensional spaces.**

I. Introduction

The canonical motion planning problem involves computing a valid path for a robot to move from a given start to a given goal state while respecting a set of physical constraints [1]–[3]. This problem is motivated by an ever-growing number of practical applications such as autonomous exploration, search-and-rescue, robotic surgery, and warehouse management just to name a few. As robots become more mobile, articulated and dexterous, it is important to find not only a feasible plan for the robot, but also one that optimizes one or more criteria for a given high-level task. The quality metric is problem-specific, and this paper will consider optimizing path length. The method described, however, is applicable to many different metrics like smoothness or obstacle clearance.

Traditional sampling-based motion planners have proven very successful in quickly computing paths for high-dimensional systems [4]–[9]. However, solutions from these planners can be unnecessarily long due to the sampling process and other heuristics used during the search. Consequently, sampling-based solutions are rarely executed without first applying some optimization. There are several existing methods for quickly shortening a path. Post-processing methods, including shortcutting [10] and the hybridization of a set of paths [11] have been shown to be highly effective at removing redundant motions from paths and forming a shorter solution from a combination of input paths, respectively. These techniques operate independently of the planner used to compute the solution and can be applied as an optional step after planning. A new class of sampling-based motion planning algorithms has also been developed [12], which are proven asymptotically optimal. These methods converge to the optimal solution path with probability one as the number of samples used during planning goes to infinity.

R. Luna, M. Moll, and L. E. Kavraki are with the Dept. of Computer Science, Rice University, Houston, TX. {rluna, mmoll, kavraki}@rice.edu

I. A. Şucan is affiliated with Willow Garage, Inc., Menlo Park, CA. isucan@willowgarage.com

This work is supported in part by NSF CCF 1018798, NSF DUE 0920721, NSF IIS 0713623, and the US Army Research Office under grant W911NF-09-1-0383. Computing resources supported in part by Cyberinfrastructure for Computational Research funded by NSF under Grant CNS 0821727.

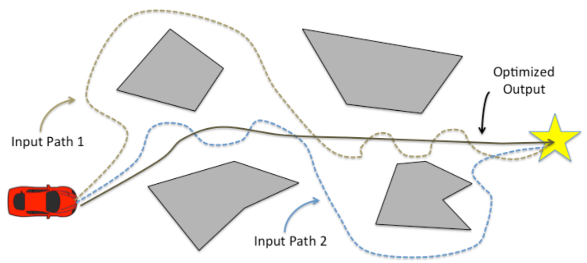


Fig. 1: An optimized solution path composed of two hybridized input paths with local shortcutting applied.

This paper addresses two questions when using sampling-based planners under time constraints: given a time budget for planning, how effective are existing post-processing methods for solution optimization across a broad spectrum of problems, and are these techniques able to compete with paths computed by algorithms which return the optimal solution given enough time? A meta-algorithm that combines the strengths of shortcutting and hybridization is presented in Section III, which allows any traditional sampling-based algorithm to operate with anytime properties. The meta-algorithm allows for the solution path to be continually optimized until computation is ceased. Experimental results show that the meta-algorithm for anytime planning can significantly reduce the path length for a given time budget across a broad spectrum of sampling-based planners. Moreover, a comparison with asymptotically optimal methods shows that the meta-algorithm is able to find solutions of similar and possibly shorter length given the same time constraints in high-dimensional spaces.

II. Previous Work

Motion planning is a fundamental problem in robotics, and has benefitted from many different perspectives in the research community. There exist a wide variety of approaches to solve this problem [1]–[3], but for high dimensional systems, sampling-based planners are typically used.

A. Sampling-based Motion Planning

Sampling-based motion planners attempt to approximate the free/valid portion of the state space through the use of sampling. Samples can be connected via a valid path to form a graph or tree structure from which a solution can be extracted.

The Probabilistic RoadMap method (PRM) [4] is the first method to employ such a scheme. The PRM begins by constructing a graph composed of valid samples which are connected via valid paths through the use of a local planner. Once the roadmap is computed, the start and goal states can be inserted into the structure, and a graph search is used to obtain a valid trajectory. Tree-based methods are also popular (e.g., EST [5], RRT [6], [7], SBL [8], KPIECE [9]), where the tree is rooted at the start state of the robot, and is grown toward the goal. The search ceases when the goal is connected to the tree. Tree-based algorithms are particularly useful for dynamic environments or non-holonomic systems.

Although sampling-based motion planners are not complete, these methods can provide probabilistic completeness, indicating that if a solution exists for a particular planning instance, the probability of discovering the solution converges to one as the number of states sampled increases to infinity.

B. Improving Sampling-Based Paths

Although sampling-based methods are able to quickly solve motion planning problems, the solutions returned by these techniques are notoriously sub-optimal due to the discrete set of states used to construct a solution path. A proof of the sub-optimality for many popular sampling-based methods is given in [12]. The sub-optimality of these methods has lent itself not only to informed heuristics for sampling-based planners, but also to path optimization techniques performed as a post-processing step to planning as well as algorithms that converge to the optimal solution.

a) Shortcutting: Shortcutting is a fast and intuitive post-processing method to significantly reduce the length of a solution path [10], [13]. This technique removes superfluous motions by selecting two points along non-consecutive segments from the path and attempting to connect these points directly via a straight path. If the direct path is valid, the original path segments connecting the two points are replaced. A general form of shortcutting is shown in Algorithm 1. This method has appeared in many variations throughout the sampling-based motion planning literature [14]–[18].

Algorithm 1 Shortcut

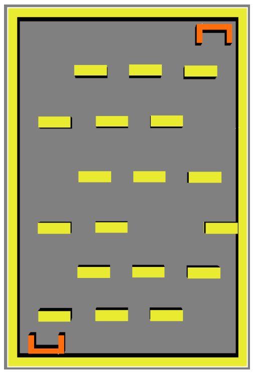
```
Input: $\pi$: path; $s$: Number of shortcutting attempts
Output: A path with potentially shorter length
1: $s' \leftarrow 0$
2: while $s' < s$ do
3:     $\widetilde{ab} \leftarrow \pi$.RandomSegment()
4:     $\overline{ab} \leftarrow$ straight path connecting endpoints of $\widetilde{ab}$
5:     if $|\overline{ab}| < |\widetilde{ab}|$ and CheckMotion($\overline{ab}$) then
6:         $\pi$.Substitute($\widetilde{ab}, \overline{ab}$)
7:     $s' \leftarrow s' + 1$
8: return $\pi$
```

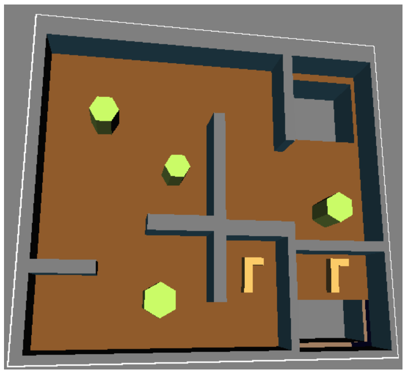
b) Hybridization: Hybridization is a technique for path optimization that combines two or more solution paths in order to form a new, hybrid path composed of the best portions of the input paths [11]. This method creates a *hybridization graph* of the input paths, where the vertices are the states of the input paths, and edges indicate valid paths between the states. This graph is initialized to the disjoint set of input paths, and valid *bridges* between the input paths are computed and inserted as additional edges into the graph. Once a set of bridges have been discovered, a classical graph search technique can be used to find a shorter, hybrid path from the *hybridization graph*.

A straightforward all-pairs computation to find the set of valid *bridges* has $O(n^2)$ complexity, and does not scale well with long input paths or a large set of paths. Two heuristic methods can be employed to improve the computation time of the method. The first takes a local approach, and only attempts to connect two paths at states that are within a prescribed distance. This is useful for situations with a large number of input paths. For scenarios where the number of input paths is bounded, a linear time dynamic programming algorithm exists to match two paths together, yielding an encoding of the cost to *bridge* any two states in the set of input paths which can be minimized.

c) Online Optimization: A wide variety of methods exists that augment existing sampling-based algorithms in order to bias the search to discover high quality paths during the planning process itself. Methods that incorporate the cost of the path into the tree expansion process [19], [20], have shown to return high quality solutions. Identifying homotopic classes within a particular environment is useful for computing a near-optimal path using roadmap based methods [21], [22], especially in environments with narrow passages. Other heuristics known to work well include visibility regions [23], reachability [24], [25], and random restarts [26]. Finally, for systems with complex dynamics, CHOMP is shown to continuously refine a potentially invalid input trajectory, eventually converging to a valid trajectory in a local minimum of the optimization space [27]. The above methods, among others, are effective in finding superior solutions compared to uninformed methods. Many of these methods, however, require extensive knowledge of the state space and/or the robotic system, or the pre-computation of expensive data structures, and the computation of these heuristics may take significantly more time than their uninformed counterparts. Moreover, these methods do not guarantee convergence to the optimal solution.

d) Asymptotically Optimal Methods: A class of sampling-based algorithms has recently been developed that converges to the true optimal solution as the number of samples increases infinitely [12]. These methods, known as asymptotically optimal sampling-based planners, opportunistically improve a solution path by employing a sophisticated heuristic to reconnect states in the roadmap/tree structure based on information gained from a new sample. In particular, RRT*, an asymptotically optimal version of the canonical RRT algorithm, has been shown to return significantly shorter paths than RRT given a specified number of samples when planning. Due to the potential for high density in roadmaps computed by these planners over long time horizons, a set of methods for near-optimal planning has been proposed [28], [29], which capture a high-quality approximation of the free space while significantly reducing the number of vertices in the graph.


(a) 2D Barriers environment


(b) 3D Cubicles environment

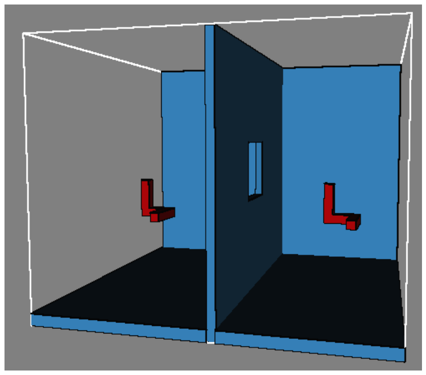

(c) 3D 'Easy' environment with 2 robots

Fig. 2: Free-flying rigid body environments. Barriers operates in SE(2), Cubicles in SE(3), and Easy has two robots in SE(3).

III. Anytime Path Shortening

This section presents a meta-algorithm that is applicable to any traditional sampling-based motion planning method, and its use gives the underlying planner an anytime quality. The shortest known solution will be continually improved through the use of shortcutting and hybridization methods, and can be retrieved at any time. Ideally, the planner used will discover many solutions during the time allotted, and by alternating between shortcutting and hybridization, it is expected that the strengths of both methods will be gained and a high quality solution will emerge. Although the technique described explicitly optimizes path length, it is possible to use a similar scheme to optimize over different criteria such as smoothness or clearance. Note that a different optimization criteria may require methods other than shortcutting or hybridization for best results (i.e., [10], [27]).

When using sampling-based planning, there are a variety of offline methods that can be used to compute a high-quality solution. Shortcutting and hybridization have the distinct advantage of being agnostic to the planner used to compute the plan, the ability to operate in very short periods of time, and can continually improve upon the quality of a path with repeated application. Shortcutting a single path segment requires just the verification of a short path using a local planner. Construction of the hybridization graph can be performed incrementally as new solutions are found, and computing the hybrid *bridges* in this graph is also verification of short paths using a local planner.

Algorithm 2 Anytime Path Shortening

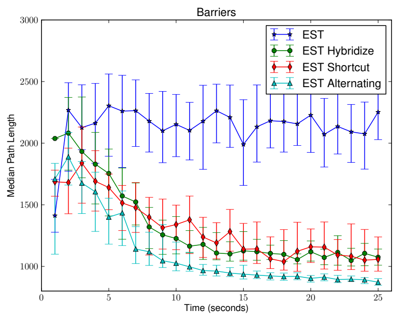
```
Input: $p$: planner; $b$: time budget; $m$: # solutions to hybridize
Output: A shortened path, if at least one path was found
1: $smooth$ = true
2: while time() $< b$ do
3:   if $p$.Plan() then
4:     if $smooth$ == true then
5:       Shortcut($p$.BestSolution())
6:     else
7:       Hybridize($m$, $p$.Solutions())
8:     $smooth$ = not $smooth$
9: return $p$.BestSolution()
```

Shortcutting and hybridization also have distinct advantages in how the path is shortened. Shortcutting allows for precise refinement of a single solution by removing redundant motions from relatively close states on the path. Hybridization, on the other hand, forms an entirely new and shorter solution path composed from the best segments of the input paths. This technique is especially helpful in discovering solutions that lie in a new homotopy class from the input paths. Note that these improvements are complementary to one another: shortcutting performs micro-level optimizations by removing small, superfluous motions from the path, and hybridization allows for macro-level improvements to a solution path by composing a new path from large portions of the existing paths. Shortcutting is unlikely to discover a new homotopy, especially in environments with narrow passages, and hybridization is not particularly suited for removing small redundancies from an otherwise high-quality path segment.

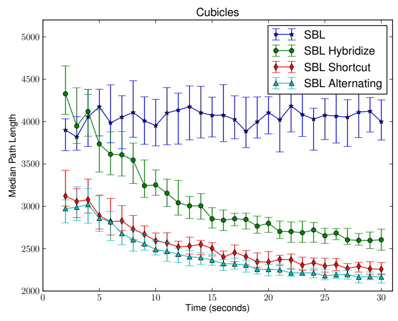
The anytime path shortening method, detailed in Algorithm 2, requires a planner, a specific time-budget, and a cap for the maximum number of solutions to hybridize. This cap is useful for large time budgets where many paths may be computed and the hybridization method begins to take significantly more time. In such a case, the best m (or random m) solutions can be hybridized.

Anytime path shortening operates over a fixed time budget, repeatedly computing solutions to the planning query (line 3). After obtaining a new solution path, the algorithm applies either shortcutting on the shortest known solution (line 5), or creates a new hybrid path from the best m solutions found (line 7). The choice of optimization technique alternates, given by the boolean flag on line 8. Once the time budget is exhausted (line 2), the shortest solution is returned (line 9).

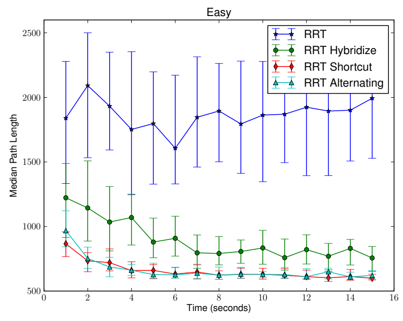
The modularity of the anytime meta-algorithm allows for flexibility in both the number and types of motion planners used, as well as the optimization techniques used to refine the solution. For general problem solving, a number of different motion planners can executed in parallel and an integer counter can be used to indicate the offline optimization scheme employed by each planner once a path is found. This broad approach is likely to capture the strength of a particular planner in a problem while mitigating the effects of another planner that is less effective for the same problem.



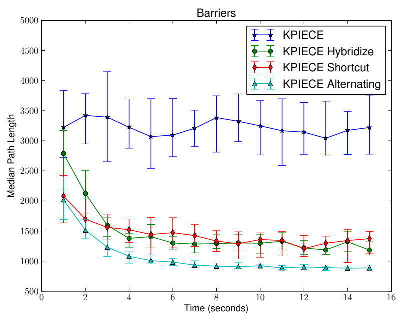
(a) Barriers: EST



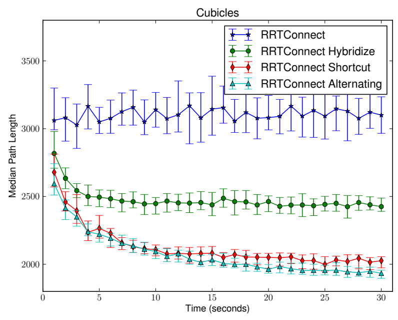
(b) Cubicles: SBL



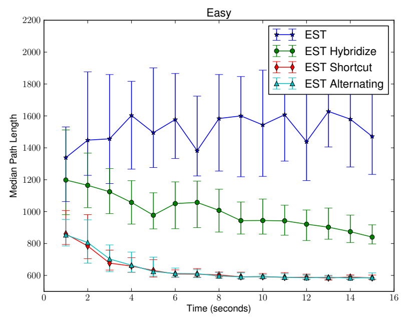
(c) 'Easy': RRT



(d) Barriers: KPIECE



(e) Cubicles: RRT-Connect



(f) 'Easy': EST

Fig. 3: Effects of anytime path shortening on a variety of planners in rigid body environments. The plot shows the median solution length with the 1st and 3rd quartiles denoted by error bars. All values are taken over 50 runs.

IV. Experiments

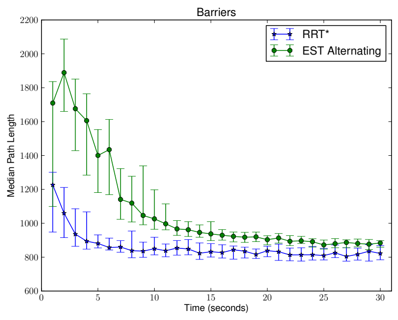
Simulated experiments presented in this section attempt to answer the questions tendered in the introduction regarding the efficacy of post-processing techniques when planning under a specific time budget, and whether these techniques can compete with algorithms that converge to the optimal solution given the same time constraints.

Implementation Details: All experiments were implemented using the Open Motion Planning Library [30], which provides implementations for all planners and optimization techniques used. In all experiments, each planner is given a specific time budget for solving the query. The total number of paths hybridized is capped at 24 due to the computational complexity of the dynamic programming algorithm for hybridization [11]. Shortcutting operates by selecting a pair of close states in the solution path and attempting to join them in the manner shown in Algorithm 1. For each call to the shortcutting procedure, the number of shortcutting attempts is capped by the total number of states in the path. To mitigate validation of very long shortcuts, the maximum length of the shortcut is capped at one-third of the total input path length; any shortcut longer than this value is not validated, and a new pair of states is selected for shortcutting.

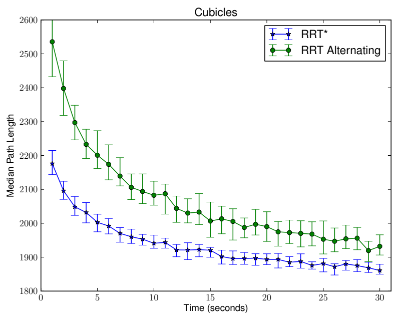
A. Rigid Body Planning

The efficacy of anytime path shortening is first evaluated in instances of free-flying rigid bodies in three environments (see Figure 2), and then compared against newer asymptotically optimal algorithms. The 2D "Barriers" and 3D "Cubicles" environments have one robot, and the 3D "Easy" environment contains two rigid bodies that must exchange states while coordinating through a narrow passage. These experiments were executed using a 2.4GHz Intel Xeon processor and 4GB memory. Anytime path shortening is compared with no optimization, as well as repeatedly applying only hybridization or only shortcutting during the same time budget.

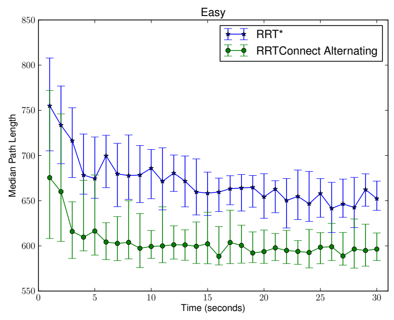
Figure 3 shows a representative sample of the benefits of anytime path shortening for a range of runtimes across a spectrum of sampling-based planners. These charts plot the median solution length along with the 1st and 3rd quartiles given a specific planning budget. It should be noted that the values in these plots are not monotonically decreasing because each data point is run independently of all others. From the figures, it is clear that the proposed anytime optimization technique tightly converges to a value at or below a similar technique that applies just shortcutting or hybridization during the same time period. In the 3-DOF "Barriers" world there are many homotopic classes that will be discovered, and anytime path shortening finds a solution on average that is 20% shorter than applying just shortcutting or hybridization. The 6-DOF "Cubicles" world has fewer homotopic classes that are likely to be discovered, and yet the anytime path shortening method finds a solution that is around 10% shorter. In the 12-DOF "Easy" world shortcutting and anytime path shortening return nearly identical solutions in this space. This can be explained by the relative sparseness of the world which contains just two symmetric homotopy classes. Finally, notice the very tight bounds on the path length with the anytime



(a) Barriers: EST



(b) Cubicles: RRT



(c) 'Easy': RRT-Connect

Fig. 4: Comparison of anytime path shortening and asymptotically optimal algorithms in rigid body planning. The plot shows the median solution length with the 1st and 3rd quartiles denoted by error bars. All values are taken over 50 runs.

path shortening algorithm. This indicates that performance is very repeatable.

Comparison with RRT:* It is interesting to compare the anytime path shortening method to asymptotically optimal planners which converge to the optimal solution as the planning time goes to infinity. This comparison is performed with the understanding that asymptotically optimal algorithms are developed for a different purpose, and provide a theoretical guarantee that has not been proven for the anytime path shortening method. These experiments compare RRT* [12] with the suite of planners seen in the worlds from Figure 2.

Figure 4 shows a representative snapshot for the performance of RRT* with respect to three other sampling-based methods using anytime path shortening. The plots chart the median solution length along with the 1st and 3rd quartiles given a specific planning budget. RRT* generally returned a shorter solution in the "Barriers" and "Cubicles" worlds, roughly 5% shorter in the limit. Surprisingly, in the 12-DOF "Easy" world the scenario is reversed, and anytime path shortening finds a solution which is consistently 5-10% shorter than that from the asymptotically optimal planner.

B. *Manipulator Planning*

Anytime path shortening is also evaluated for instances of manipulator planning. Planning queries for the 7-DOF arm of a PR2 robot from Willow Garage are performed using the ROS infrastructure and MoveIt! planning package [31] in conjunction with OMPL [30]. Three scenes were constructed; these can be seen in Figure 5. The first is an empty environment with the exception of a post blocking some motion of the right arm. The second scene is a table environment where the motion of the right arm must be planned around an obstacle on top of the table. The third scene is a cluttered environment with many small obstacles within range of the PR2's arm. All experiments were performed on a quad-core Intel Core 2 Duo 2.4 GHz with 4GB memory.

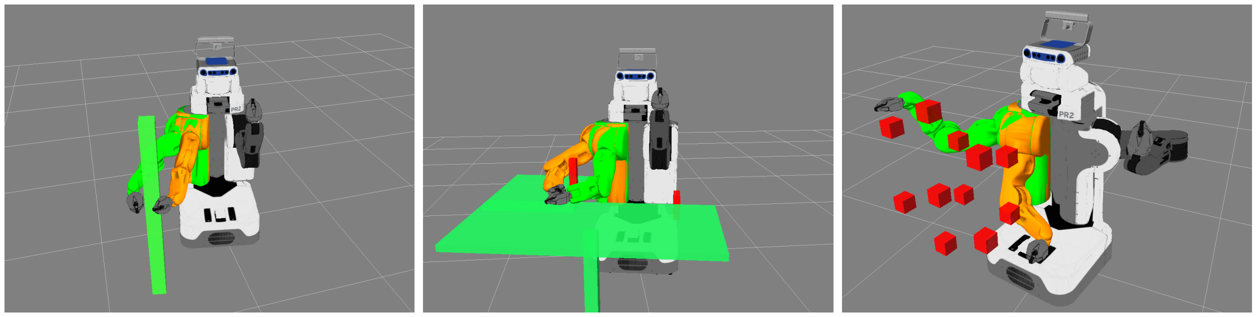
Similar to the rigid body experiments above, Figure 6(a)-(c) shows the effects of anytime path shortening for a representative sample of planners. The plots show that shortcutting dominates in the "Post" and "Table" scenes, which can be explained by the relative sparseness of environment coupled with just one homotopy class for the solution. In the "Clutter" scene, pure hybridization achieves a solution length about 15% longer on average when compared to shortcutting or the anytime path shortening method. The size of the obstacles in this scene combined with the joint limits of the PR2 greatly affects the ability for solutions of other homotopic classes to be discovered. This explains why shortcutting and the anytime path shortening method find very similar solutions for all three of these scenes, and further investigation of the effects of anytime path shortening is left as future work.

The manipulator planning scenes are also compared to the RRT* planner which converges to the optimal solution given an infinite time budget. Figure 6(d)-(f) plots the median solution length returned by RRT* to a representative series of planners using the anytime path shortening technique. Surprisingly, RRT* finds solutions ranging from 20% longer in the "Clutter" world to nearly 50% longer in the "Post" environment when compared to traditional sampling-based methods using anytime path shortening. Also, note the tight convergence for the anytime path shortening method as the time budget increases. Over a longer horizon, the solution returned by anytime path shortening becomes more predictable.

In addition to solution length, it is noteworthy to point out the rate at which the planners failed to find any solution at all in these experiments, shown in Figure 7. The RRT* planner exhibited a significant rate of failure in these experiments, particularly in the "Post" and "Clutter" environments where rates of 50% or higher were seen even at longest planning budgets. On the other hand, traditional sampling-based methods had very few failures, and those that were seen occurred during the shortest time budgets.

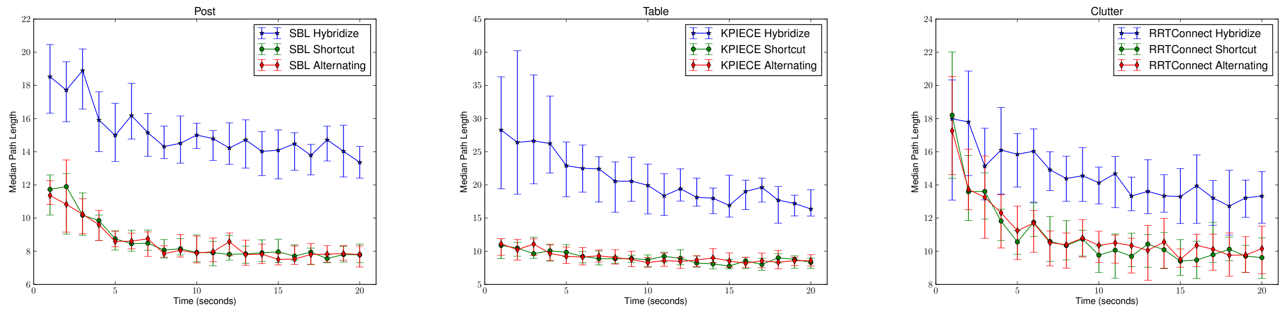
V. Discussion

This paper presented a meta-algorithm for converting any sampling-based motion planner into one with anytime properties that continually shortens the solution through a combination of existing optimization techniques. The meta-algorithm shows improvement over a wide spectrum of sampling-based planners given a specific planning time. In addition to improvements in path length, greater repeatability is also observed. Moreover, in high-dimensional scenarios the anytime technique exhibited significant improvements in

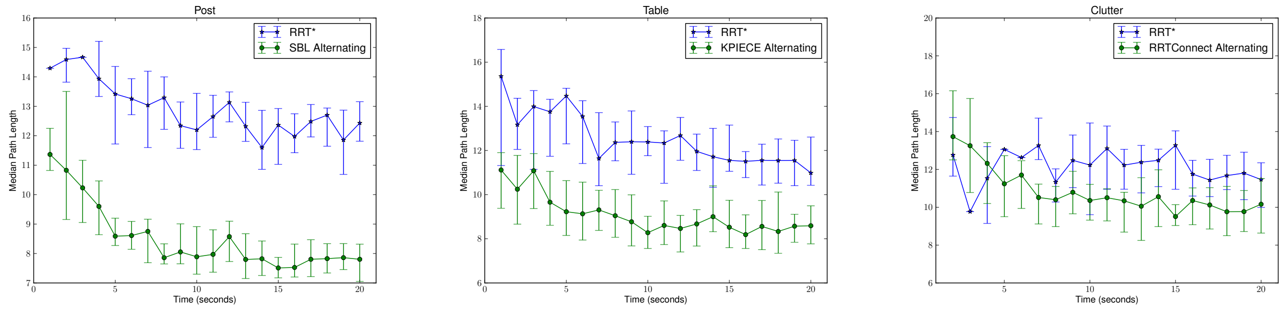


(a) Post (b) Table (c) Clutter

Fig. 5: Manipulator scenes for the 7-DOF arm of the PR2. The start pose is shown in green, and the goal pose is orange.



(a) Post: SBL (b) Table: KPIECE (c) Clutter: RRT-Connect



(d) Post: SBL (e) Table: KPIECE (f) Clutter: RRT-Connect

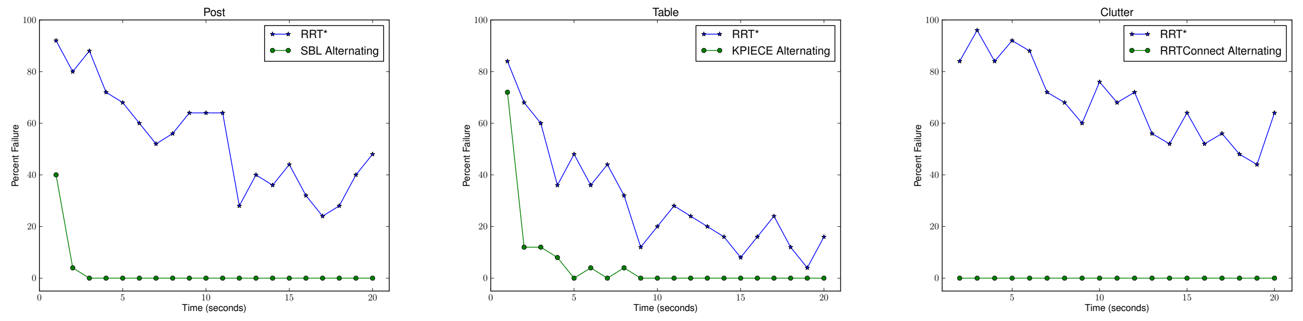
Fig. 6: Effects of anytime path shortening and a comparison with asymptotically optimal algorithms in manipulator planning. Plots show the median solution length with the 1st and 3rd quartiles denoted by error bars. All values are taken over 25 runs.

solution length against a planner which returns the optimal solution given enough planning time.

This work is not meant to directly compare any two sampling-based algorithms, and such a comparison is outside the scope of this paper. Previous work has shown that one particular planner may be better suited for a specific robot or environment. Furthermore, many sampling-based algorithms provide parameters and other metrics that can be fine tuned for a particular planning context. However, an interesting insight into the poor behavior of the asymptotically optimal RRT* in high dimensional spaces may be attributed to the well-known difficulty of defining a good distance metric in complex spaces to take advantage of the *Voronoi bias* property of RRT [32], [33]. The study presented in this work shows that offline heuristic techniques for solution optimization can outperform specialized planners which compute the globally optimal path in certain scenarios.

It should also be noted that utilizing asymptotically optimal planners inside the anytime path shortening framework is non-trivial. The anytime framework relies on a planner's ability to quickly return paths while spending the remaining time budget optimizing the output. Asymptotically optimal planners do not necessarily halt once a path to the goal has been computed, but rather continually optimize the solution until a set time budget or number of states has been expanded.

There are several interesting avenues for further expansion of the comparison to online and offline optimization of motion plans. In particular, trajectories for non-holonomic systems cannot be easily optimized using shortcutting or hybridization. Techniques like CHOMP [27] attempt to smoothly deform an invalid trajectory into one which is valid using a gradient descent-like approach over a cost function, but generating high-quality trajectories for non-holonomic systems is still an active field of research.



(a) Post: SBL/RRT* (b) Table: KPIECE/RRT* (c) Clutter: RRT-Connect/RRT*

Fig. 7: The percentage of experiments that failed to find any solution to the given query within the given time budget. Values are out of 25 possible attempts.

References

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

[2] H. Choset, W. Burgard, S. Hutchinson, G. Kantor, L. E. Kavraki, K. Lynch, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, June 2005.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge Univ. Press, 2006.

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

[5] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Intl. J. of Computational Geometry and Applications*, vol. 9, no. 4/5, pp. 495–512, 1999.

[6] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Intl. J. of Robotics Research*, vol. 17, no. 5, pp. 378–400, 2001.

[7] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Intl. Conf. on Robotics and Automation*, April 2000, pp. 995–1001.

[8] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *The Tenth International Symposium on Robotics Research*, 2001, pp. 403–417.

[9] I. Şucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 116–131, 2012.

[10] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *Intl. J. of Robotics Research*, vol. 26, no. 8, pp. 845–863, August 2007.

[11] B. Raveh, A. Enosh, and D. Halperin, "A little more, a lot better: Improving path quality by a path-merging algorithm," *IEEE Trans. on Robotics*, vol. 27, no. 2, pp. 365–371, April 2011.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Intl. J. of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[13] S. Berchtold and B. Glavina, "A scalable optimizer for automatically generated manipulator motions," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Sept 1994, pp. 1796–1802.

[14] D. Hsu, J.-C. Latombe, and S. Sorkin, "Placing a robot manipulator amid obstacles for optimized execution," in *IEEE International Symposium on Assembly and Task*, July 1999, pp. 280–285.

[15] C. Geem, T. Siméon, J.-P. Laumond, J.-L. Bouchet, and J.-F. Rit, "Mobility analysis for feasibility studies in cad models of industrial environments," in *IEEE Intl. Conf. on Robotics and Automation*, May 1999, pp. 1770–1775.

[16] P. Isto, "Constructing probabilistic roadmaps with powerful local planning and path optimization," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Sept 2002, pp. 2323–2328.

[17] 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*, Sept 2003, pp. 2424–2429.

[18] D. Nieuwenhuisen and M. H. Overmars, "Useful cycles in probabilistic roadmap graphs," in *IEEE Intl. Conf. on Robotics and Automation*, April 2004, pp. 446–452.

[19] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Oct 2003, pp. 1178–1183.

[20] D. Ferguson and A. Stentz, "Anytime RRTs," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2006, pp. 5369–5375.

[21] E. Schmitzberger, J. 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*, Oct 2002, pp. 2317–2322.

[22] L. Jaillet and T. Siméon, "Path deformation roadmaps," in *Workshop on the Algorithmic Foundations of Robotics*, July 2006.

[23] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Journal of Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[24] 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*, 2006, pp. 4355–4361.

[25] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *IEEE Intl. Conf. on Robotics and Automation*, 2009, pp. 2859–2865.

[26] N. A. Wedge and M. S. Branicky, "On heavy-tailed runtimes and restarts in rapidly-exploring random trees," in *Twenty-Third AAAI Conference on Artificial Intelligence*, July 2008, pp. 127–133.

[27] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *IEEE Intl. Conf. on Robotics and Automation*, May 2009.

[28] J. D. Marble and K. E. Bekris, "Towards small asymptotically near-optimal roadmaps," in *IEEE Intl. Conf. on Robotics and Automation*, May 2012, pp. 2557–2562.

[29] A. Dobson, A. Krontiris, and K. E. Bekris, "Sparse roadmap spanners," in *Workshop on the Algorithmic Foundations of Robotics*, 2012.

[30] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, Dec. 2012, http://ompl.kavrakilab.org.

[31] S. Chitta, I. A. Şucan, and S. Cousins, "MoveIt! [ROS Topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[32] P. Cheng and S. M. LaValle, "Reducing metric sensitivity in randomized trajectory design," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, vol. 1, 2001, pp. 43–48.

[33] Y. Li and K. E. Bekris, "Learning approximate cost-to-go metrics to improve sampling-based motion planning," in *IEEE Intl. Conf. on Robotics and Automation*, May 2011, pp. 4196–4201.