An empirical study of single-query motion planning for grasp execution

Meijer, Jonathan; Lei, Qujiang; Wisse, Martijn

DOI
10.1109/AIM.2017.8014187

Publication date
2017

Document Version
Accepted author manuscript

Published in
Proceedings 2017 IEEE International Conference on Advanced Intelligent Mechatronics

An Empirical Study of Single-Query Motion Planning for Grasp Execution

Jonathan Meijer, Qujiang Lei, Martijn Wisse

***Abstract*—This paper identifies high-performing OMPL planners, available in MoveIt!, when carrying out several grasp executions with a UR5 manipulator. Simultaneously, this paper presents useful benchmark data. The single-query performance of the planners was measured by means of solved runs, computing time and path length. Based on the results, recommendations are made for planner choice that shows high performance.**

I. Introduction

MoveIt! [1] is widely used for robot manipulation within ROS (Robot Operating System). MoveIt! comes with the Open Motion Planning Library (OMPL) [2] plugin. This lets the user easily choose state-of-the-art sampling-based motion planners from the OMPL library. Currently, 23 motion planning algorithms of OMPL are configured to use in MoveIt!. Recommendations for picking a motion planning algorithm is not given. The Planner Arena [3] is created to help users determine which planner suits a given motion planning problem. However, none of the problems resemble the use of a manipulator performing a grasp execution.

This paper aims to provide insight in choosing the right planner(s) for performing grasp executions by conducting an empirical study on the available motion planners of OMPL in MoveIt!. We choose to only investigate the single-query performance of the available sampling-based motion planners, included multi-query planners are being used as single-query planners. The performance of such planners depends on the configuration space of the robot, in particular, motion planning through narrow passages can cause issues [4]. For this paper, four grasp execution motions are defined in the MoveIt! Benchmark environment to discover the behavior of the planners. The manipulator that will be used in the benchmarking is the UR5 robot due to its universal use. To resemble the real-world grasping problem, we fitted the manipulator with a virtual camera and gripper.

The performance of the planners is measured in terms of solved runs, computing time and path length. Solved runs can be noted as a percentage of the total motion planning runs that finish correctly, barplots are used to visualize the difference. For every run, the total computing time and path length can change due to the randomization in sampling-based motion planners. Boxplots and tables are used to analyze the planners with respect to computing time and path length. Performance depends on the users need, right planners for one performance measure can be wrong planners for a different performance measure. By looking at each measure, we can discuss on the preferred planner choice.

*The work leading to these results has received funding from the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement n 609206.

All authors are with the TU Delft Robotics Institute, Delft University of Technology, 2628 CD, Delft, The Netherlands. Email address of Jonathan Meijer is J.G.J.Meijer@student.tudelft.nl. Email addresses of Qujiang Lei and Martijn Wisse are q.lei, m.wisse@tudelft.nl.

TABLE I: Summary of available planners of OMPL in MoveIt!

Planner name	Optimizing planners	Time-invariant goal
SBL [5]		✓
EST [6]		✓
BiEST [6]		✓
ProjEST [6]		✓
KPIECE [7]		✓
BKPIECE [7]		✓
LBKPIECE [7][8]		✓
RRT [9]		✓
RRTConnect [10]		✓
PDST [11]		✓
STRIDE [12]		✓
PRM [13]		
LazyPRM [8]		
RRTstar [14]	✓	
PRMstar [13][14]	✓	
LazyPRMstar [8][14]	✓	
FMT [15]	✓	✓
BFMT [16]	✓	✓
LBTRRT [17]	✓	✓
TRRT [18]	✓	✓
BiTRRT [19]	✓	✓
SPARS [20]	✓	
SPARStwo [21]	✓	

Through a quick survey of the available planners of OMPL in MoveIt!, several observations can be made on how they handle motion problems. The comparison (Tab. I) shows that the promise of using FMT, BFMT, LBTRRT, TRRT and BiTRRT. These planners have an optimizing step and stop once an optimized path is found.

II. Background

A. Software

The open-source Robot Operation System is a suite of software libraries that help create robot applications. ROS was created to encourage collaborative robotics software development. MoveIt! serves as a framework in ROS to help with the manipulation of robotic hardware. Within MoveIt! several motion planner libraries can be added to perform motion planning for the specified robot. OMPL is a well-integrated motion planner library and is the default library for MoveIt!. It houses state-of-the-art sampling-based motion planners. OMPL itself only implements the basic primitives of sampling-based motion planning. MoveIt! configures OMPL

and provides the back-end for OMPL to work with motion planning problems.

When executed through MoveIt!, OMPL creates a path to solve the motion planning problem. By default, OMPL tries to perform path simplification. These are routines that shorten the path. The smoothness of the path may not be affected by this simplification.

B. Overview planning algorithms

Sampling-based motion planners are proven to be probabilistically complete [13], which implicates that the probability of not finding a feasible path in an unbounded setting approaches zero. For this reason, sampling-based motion planners are widely used to find feasible paths in high-dimensional and geometrically constraint environments. Optimizing planners can refrain from potential high-cost paths and rough motions [14]. However, computational effort for finding an optimized path is increased.

Among the 23 motion planners in Tab. I, six can be considered as multi-query planning methods. A common multi-query planning method is the Probabilistic RoadMap (PRM) [13]. The planner attempts to find a path in a constructed roadmap. The construction of the roadmap is executed by sampling valid nodes (configuration states) in the configuration space. These nodes are connected to other nearby nodes by edges (path segments). In OMPL, the roadmap construction is finished when a certain time limit is reached. Afterward, a simple graph search (query) can be performed on the roadmap to find a path between the start and goal node. Because the algorithm covers the total configuration space with a roadmap, *multiple queries* can be started to find a path with different a start and goal node.

In MoveIt!, three variants of the PRM planner are available for use. The LazyPRM [8] planner initially does not check for valid states when sampling nodes for roadmap construction. Once a path has been found from between start and goal node, collision checking is performed along the nodes and edges of the roadmap. Invalid nodes and edges are removed and a new graph search is attempted. This process is repeated until a feasible path is found. PRMstar [14] is the asymptotically optimal variant of the PRM planner. It rewires nodes to other near nodes if this is beneficial to the cost towards the node. An asymptotically optimal path is found if there is a great number of nodes. LazyPRMstar [14] combines the LazyPRM and PRMstar.

In addition to the PRM planner and its variants, OMPL has two more multi-query planners, SPARS and SPARStwo. They are similar to PRMstar but adds another sparse subgraph. This subgraph is an asymptotically optimal roadmap that houses nodes which resemble multiple nodes in a dense graph. Therefore less computing memory is needed to store the asymptotically optimal roadmap. SPARStwo is different since it has an infinite iteration loop.

The remaining 17 planners in Tab. I are considered as single-query planning methods. These create a roadmap every time a new planning query has to be determined. A common single-query planner is the Rapidly Exploring Random Tree (RRT) method [9]. It grows one tree (mono-directional) from the initial configuration state in the direction of the unexplored areas of the bounded free space. This is realized by randomly sampling nodes in the free space, sampled nodes that can be are within a certain distance of tree nodes are added to the tree by edges. The process of adding nodes and edges is repeated until the tree reaches the goal node. The *goal bias* parameter in this planner specifies the probability of choosing the goal configuration as a sample rather than a random sample.

The RRTConnect method [10] is a bi-directional version of the RRT method, meaning that two trees are grown. Two processes of RRT are started, one in the start node and one in the goal node. At every iteration or edge addition, it is checked whether the trees can be connected to each other. A path that solves the motion planning problem, is found if these trees can be connected. The near-optimal variant of RRT, RRTstar [14], checks whether the new sampled node can be connected to other near nodes so that the state space is more locally refined. The RRTstar removes the connections of the new sample that are not beneficial towards the cost of the path, like PRMstar. When the number of nodes is big enough, it can result in an asymptotically optimal path from the start-to-goal node. As shown in Tab. I, the RRTstar goal is time-invariant. It keeps trying to optimize the trees by adding new nodes until specified time limit is met.

Lower Bound Tree-RRT (LBT-RRT) [17] is an asymptotically optimal planner and uses a so-called lower bound graph which is an auxiliary graph. To maintain the tree, a similar method as RRTstar is used. Transition-based RRT or TRRT [18] is a combination of the RRT method and a stochastic optimization method for global minima. It performs transition tests to accept new states to the tree. The algorithm computes an optimized path that is not tied to a time limit, unlike RRTstar. The Bi-TRRT [19] is a bi-directional version of this planner.

The EST method [6] stands for Expansive Space Trees. Other than RRT, EST tries to determine the direction of the tree by looking at neighboring nodes. The tree will grow in the direction of the less explored space. Bi-directional EST (BiEST), based on [6], grows two trees like RRTConnect. Projection EST (ProjEST), also based on [6], detects the less explored area of the configuration space by using a grid. This grid serves as a projection of the state space. Single-query Bi-directional probabilistic roadmap planner with Lazy collision checking, also called SBL, grows two trees. The trees expand in the same manner as EST. Due to its lazy collision checking it will determine if a path is valid after the two trees are connected. It deletes nodes and edges of the path that are not valid, similar to LazyPRM.

KPIECE (Kinodynamic motion Planning by Interior-Exterior Cell Exploration) [7] is a tree-based planner that uses layers of discretization to help estimate the coverage of the state space. The OMPL implementation only uses one layer. OMPL incorporates a bi-directional variant called BKPIECE and a variant which incorporates lazy collision checking, this

is the LBKPIECE.

Fast Marching Tree (FMT) [15] is an asymptotically optimal planner which marches a tree forward in the cost-to-come space on a specified amount of samples. The BFMT [16] planner is a bi-directional variant of this planner.

PDST (Path-Directed Subdivision Tree) [11] represents samples as path segments instead of configuration states. It uses non-uniform subdivisions to explore the state space.

STRIDE (Search Tree with Resolution Independent Density Estimation) [12] uses a Geometric Nearneighbor Access Tree (GNAT) to sample the density of the configuration space. This information helps to guide the planner into the less explored area.

III. Problem Formulation

The available planners consist of non-optimizing and optimizing planners. The problem formulation follows the work of Karaman and Frazzoli [14]. Non-optimizing planners attempt to find a feasible path in the bounded d-dimensional configuration space $C = [0, 1]^d$. The free configuration space is defined by $C_{\text{free}} = cl(C \setminus C_{\text{obs}})$, in which $cl(\cdot)$ denotes the closure of a set and in which C_{obs} denotes the obstacle space. A path p is called feasible when:

$$p(0) = x_{\text{init}},\ p(1) = x_{\text{goal}} \qquad (1)$$
$$p(x) \in C_{\text{free}} \text{ for all } x \in [0, 1]$$

Optimizing planners that are given a motion planning problem $(C_{\text{free}}, x_{\text{init}}, x_{\text{goal}})$ and a cost function c, find a optimized path p^* such that:

$$c(p^*) = \min\{c(p) : p \textit{ is feasible } \} \qquad (2)$$

Problem implementation. Non-optimizing planners are asked to produce feasible paths with a maximum computing time of 3s and 10s. Optimizing planners are asked to produce an optimized path within a maximum computing time of 3s and 10s. Path simplification by OMPL is turned on.

Performance metric. Solved runs, computing time and path length are used as metric in our experiments. We analyze the measures individually to provide the best performing planners in each one of the measures. Solved runs is analyzed in terms of percentage of total runs of the planner resulting in feasible paths, higher performance is considered for higher solved runs. Total computing time is measured for the time it takes for planners to produce feasible or optimized paths with path simplification, a shorter time is considered as higher performance. Moreover, planners with a small standard deviation from the average computing time and small interquartile range are considered as better performance. Path length is measured by the length of the sum of motions for a produced path. Shorter lengths are considered as higher performance. Again, planners with a small standard deviation from the average path length and small interquartile range are considered as better performance.

Parameters. In MoveIt! and OMPL parameters can be set to increase the performance of the planners. To choose them, we conducted an iterative process in which we investigated the different parameter settings for each planner by increasing and decreasing the default settings by a factor of two. Parameter tuning was conducted again in the direction performance increase was noticed until no better performance was achieved. These new parameter values were then set before conducting the benchmarks.

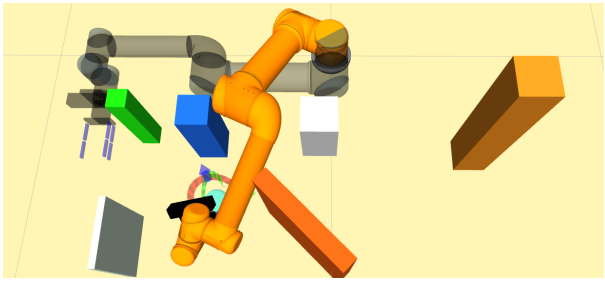


Fig. 1: Benchmark 1: Grasp between obstacles

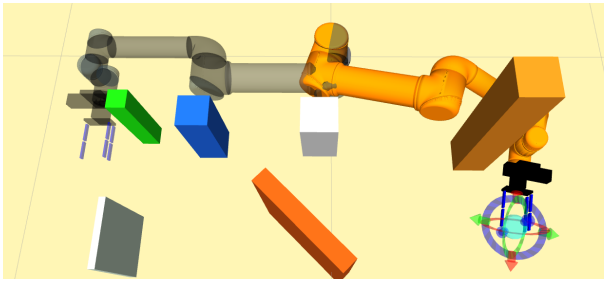


Fig. 2: Benchmark 2: Long motion

IV. Defined Motion Planning Problems

Four grasp execution motions have been defined to measure the performance of the planners.

A. Grasp between obstacles

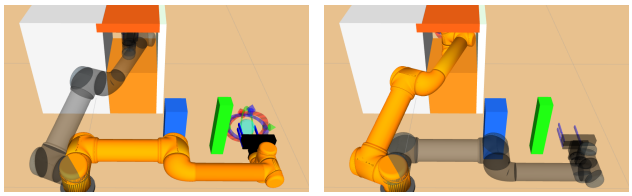
Benchmark 1 has its end-effector goal placed in a such a way that the manipulator has to move through a narrow passage, shown in Fig. 1. This benchmark uses an environment that resembles a table with obstacles. To be able to move through the narrow passage the UR5 robot needs to be in a specific configuration due to its geometry, this decreases the free configuration space near the goal configuration.

B. Simple motion

Benchmark 2 has its end-effector goal placed at the other side of the scene, shown in Fig. 2. It operates in the same environment as benchmark 1. The end-effector has to be displaced 1.5m in order to reach the goal. This motion planning problem can be solved by mainly actuating the shoulder lift joint. To actuate less joints, using optimizing planners could be beneficial.

C. Place motion

In benchmark 3, initial end-effector position is located at the end of a shelf box, shown in Fig. 3a. This benchmark uses an environment that houses a simplified shelf box and obstacles placed on a flat surface. The goal position is the



(a) Place motion (b) Pick motion

Fig. 3: Benchmark 3 and 4: a, b respectively

initial configuration used in benchmark 1 and 2 (orange state in figure). The motion problem starts in a narrow passage (transparent state in figure), the goal is situated in a less constrained space.

D. Pick motion

Benchmark 4 is attempting to resemble a picking motion from a narrow shelf box, shown in Fig. 3b. The environment is identical to benchmark 3. The goal of the problem is to be in a specific gripper orientation at the end of this shelf. The planner will have to produce a motion plan with high accuracy to reach the end of the shelf. The motion problem starts in a less constrained space (transparent state in figure), the goal is situated in a narrow passage (orange state in figure).

V. Parameter selection

Parameters can be set to improve the performance of the planners. In this section, the parameter selection is presented.

While conducting parameter selections for LBTRRT it was found that this planner is behaving unreliable in our setup. We tested all parameter combinations for this planner when conducting various motion planning which resulted in crashes. So we are unable to provide benchmark data for this particular planner.

A. Global planner parameter

There is one parameter that affects all planners. This is the distance parameter *longest_valid_segment_fraction*. The parameter is called when the planner checks for collisions between two nodes. Collision detection is not checked for the motion if the distance between the nodes is within the parameter value. In narrow passages and corners, this parameter can be critical. The parameter is set in meters and by default has a value of 0.005m. After conducting experiments with lower values, it was found that reducing this parameter did not have an immediate effect on the solved runs for the various benchmark problems.

B. Planner specific parameters

The majority of planners (20 of 23) have their own parameters. For the benchmarks, each parameter was set to values that benefit one or more performance measures, these values are noted in Tab. II.

C. Robot

The UR5 robot that will be used has two joint limit settings for each joint, π and 2π. Validating by means of simple motion planning experiments it was found that setting the joint limits to π resulted in favorable performance for all the performance measures.

VI. Results

A. Methodology

The benchmarking experiments are performed using one thread on a system with an Intel i5 2.70GHz processor and 8Gb of memory. To give reliable data on the solved runs, computing time and path length, each algorithm was run 30 times for the given motion planning problem. The algorithms were given a maximum computing time of 3s and 10s to show the effect of time on the algorithms for which the goal is not time-invariant (shown in Tab. I). The times are kept low since for most robotics applications results are required quickly. Specifically for grasping approaches that try to find grasps in a short amount of time. Planners, where path simplification was not performed by OMPL, have been marked with a * behind the planner name.

B. Plots and Tables

Results of benchmark 1 are shown in Fig. 4 and Tab. III. The motion planning problem affects planners EST, RRT, RRTstar, TRRT and SPARStwo since they were not able to solve all the runs with a percentage higher than 80% with a maximum computing time of 3s and 10s. With exception of SPARStwo, these are mono-directional planners. SBL, BiEST, KPIECE, BKPIECE and LBKPIECE compute valid paths in a computing time shorter than 0.5s. RRTConnect is the fastest planner and BiTRRT is the fastest optimizing planner. SBL has the lowest average path length with a small standard deviation. For solved runs higher than 80%, planners SBL, KPIECE, and LBKPIECE are able to plan paths of similar lengths. For optimizing planners, BiTRRT has the lowest median path length. TRRT has the lowest average path length and standard deviation. Selecting a higher limit for computing time showed shorter paths for planners RRTstar, PRMstar and LazyPRMstar, due to the optimization step. Path simplification contributes to shorter paths.

Results of benchmark 2 are shown in Fig. 5 and Tab. IV. RRTstar, TRRT and SPARStwo have lower solved runs compared to the other planner algorithms. SBL, BiEST, BKPIECE, LBKPIECE, RRTConnect and BiTRRT compute paths in under 0.1s, all being bi-directional planners. BiTRRT is the fastest optimizing planner. SBL and BiTRRT have the shortest paths. The planners that keep sampling the configuration space or optimizing the path until the maximum computing time is reached see improved performance with respect to path length. However, compared to non-optimizing planners

Results of benchmark 3 are shown in Fig. 6 and Tab. V. None of the multi-query planners are able to find feasible paths. Increased computing effort is needed to cover the total

TABLE II: Specified planner parameters

SBL	EST	BiEST	ProjEST	RRT	RRTConnect	PRM	LazyPRM	RRTstar	
range: .3125	range: .625 goal bias: .05	range: 0	range: .625 goal bias: .05	range: 0 goal bias: .05	range: .3125	max n.n.: 10	range: .3125	range: 0 goal bias: .05 delay c.c.: 0	
KPIECE	**BKPIECE**	**LBKPIECE**	**STRIDE**	**FMT**	**BFMT**	**TRRT**	**BiTRRT**	**SPARS**	**SPARStwo**
range: .625 goal bias: .05 border frac.: .9 failed e.s.f.: .5 min.v.p.frac.: .5	range: .3125 border frac.: .9 failed e.s.f.: .5 min.v.p.frac.: .5	range: .3125 border frac.: .9 min.v.p.frac.: .5	range: .625 goal bias: .05 use proj.dist.: 0 degree: 8 max degree: 12 min degree: 6 max p.p. leaf: 3 est. dim.: 0 min.v.p.frac.: .1	samples: 1000 rad. mult.: 1.05 nearest k: 1 cache cc: 1 heuristics: 1 extended fmt: 1	samples: 1000 rad. mult.: 1.05 nearest k: 1 balanced: 1 optimality: 0 cache cc: 1 heuristics: 1 extended fmt: 1	range: 1.25 goal bias: .05 max s. f.: 10 temp c. fact.: 2 m.temp.: 1e-10 i.temp.: 1e-6 f. threshold: 0 f.NodeRatio: .1 k constant: 0	range: 1.25 temp c. fact.: .2 init temp.: 50 f. threshold: 0 f. n. ratio: .1 cost.thres.: 5e4	str. factor: 2.6 sp. d. frac.: .25 d. d. frac.: .001 max fails: 1000	str. factor: 3 sp. d. frac.: .25 d. d. frac.: .001 max fails: 5000

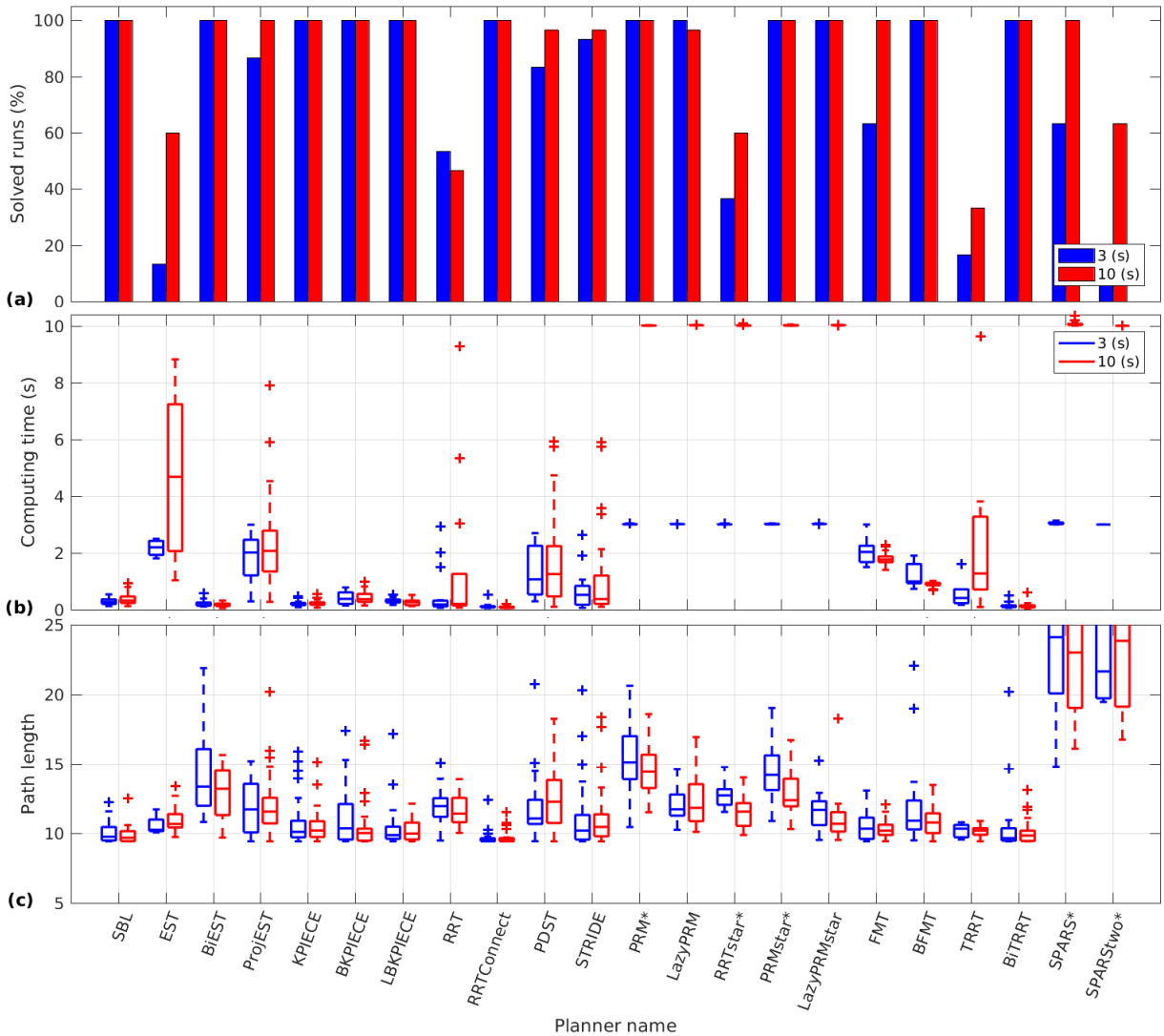


Fig. 4: Results for benchmark 1 for 3s and 10s maximum computing time. a. Solved runs (higher is better), b. Computing time (lower is better, small interquartile range is better), c. Path length (lower is better, small interquartile range is better).

free configuration space with these planners. Of the single-query planners BiEST, RRT, RRTstar and FMT are not able to reach a high level of solved runs. Indicating that these planners are not fast enough to have a proper coverage of the free configuration space. BKPIECE and RRTConnect reach 100% solved runs for 3s maximum computing time. SBL, EST, ProjEST, KPIECE, LBKPIECE, PDST, STRIDE, TRRT and BiTRRT perform better with respect to solved runs when the maximum computing time is set to 10s. RRTConnect is the fastest planner, TRRT is the fastest optimizing planner with solved runs higher than 50%. With exception of RRTConnect, the single-directional planner variants are able to plan a shorter path length compared to the bi-directional planner variant. These planners propagate a path out of a narrow passage and with the use of goal bias shorter paths can be obtained.

Results of benchmark 4 are shown in Fig. 7 and Tab. VI.

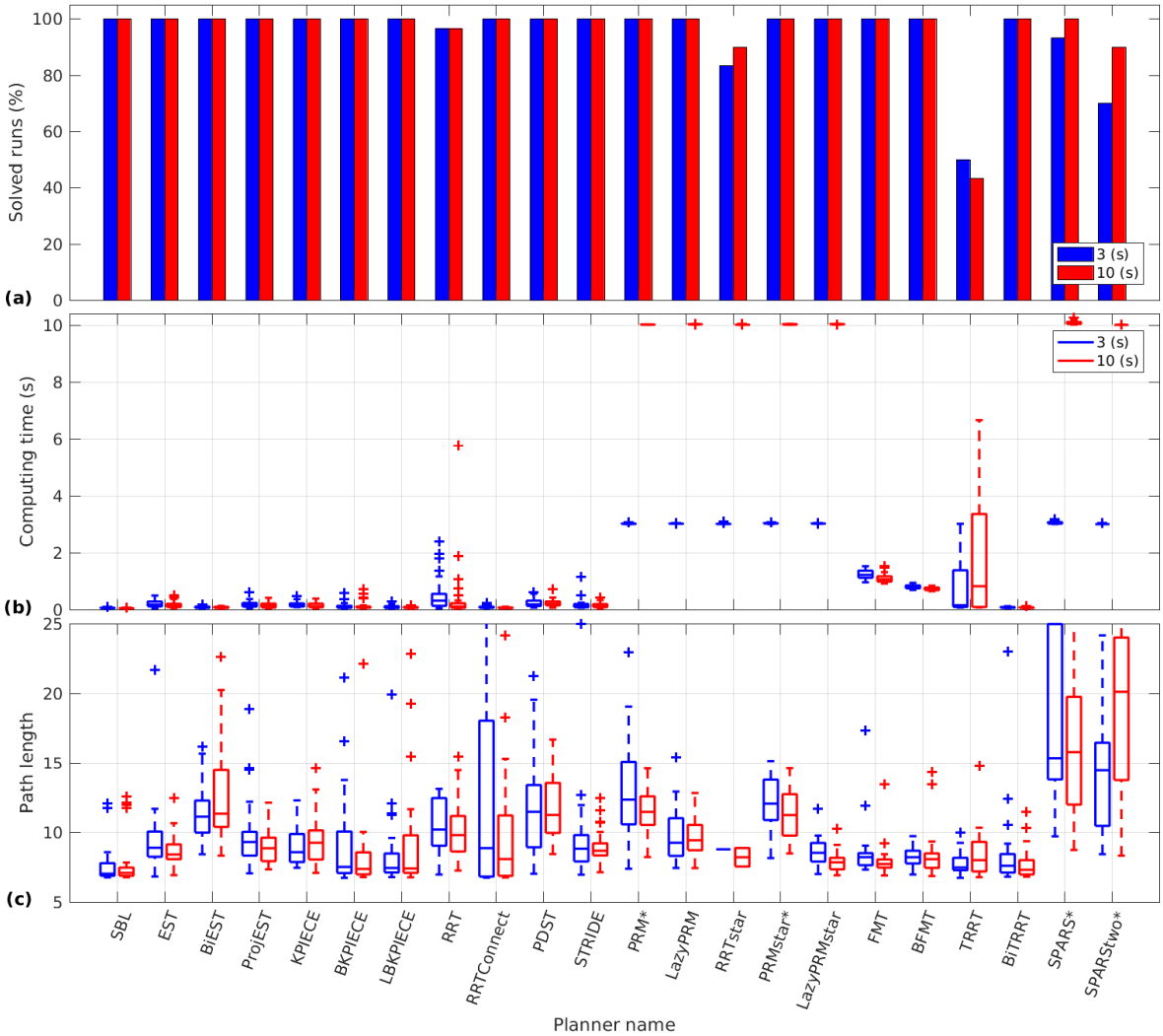


Fig. 5: Results for benchmark 2 for 3s and 10s maximum computing time. a. Solved runs (higher is better), b. Computing time (lower is better, small interquartile range is better), c. Path length (lower is better, small interquartile range is better).

5 of the 22 planners were able to compute paths with solved runs higher than 50%. These are all bi-directional planners, motion planning with these planners are also started in the goal configuration. These planners provide high solved runs for max 10s computing time. KPIECE and RRTConnect are the fastest performing planners. BiTRRT has the shortest path planning length. RRTConnect shows significant performance increase for path length with a higher maximum computing time.

C. Discussion

From the results, observations are made and discussed.

Solved runs. The planners RRTstar, TRRT and SPARStwo show consistent lower solved runs for all the benchmarks, making them less desirable to use for the grasp executions we presented. SBL, LBKPIECE and BiTRRT have high solved runs for a maximum computing time of 10s in all benchmarks. When high solved runs has to be achieved in a shorter time, BKPIECE and RRTConnect are the best choices when performing varied grasp executions.

TABLE III: Average values for benchmark 1

Planner name	Max. 3s computing time		Max. 10s computing time	
	Time (s)	Path length	Time (s)	Path length
SBL	0.29 (0.11)	10.07 (0.74)	0.37 (0.18)	9.90 (0.63)
EST	2.18 (0.31)	10.58 (0.77)	4.65 (2.55)	11.03 (1.01)
BiEST	0.21 (0.10)	14.81 (5.19)	0.18 (0.07)	13.32 (3.14)
ProjEST	1.83 (0.86)	11.82 (1.81)	2.37 (1.57)	12.13 (2.19)
KPIECE	0.20 (0.09)	10.89 (1.80)	0.22 (0.10)	10.55 (1.28)
BKPIECE	0.42 (0.21)	10.94 (1.86)	0.42 (0.21)	10.56 (1.82)
LBKPIECE	0.30 (0.08)	10.41 (1.53)	0.26 (0.11)	12.30 (7.16)
RRT	0.54 (0.84)	11.93 (1.41)	1.48 (2.71)	11.62 (1.14)
RRTConnect	0.11 (0.08)	12.52 (15.68)	0.09 (0.03)	11.89 (9.10)
PDST	1.37 (0.87)	11.96 (2.35)	1.68 (1.61)	12.37 (2.15)
STRIDE	0.59 (0.57)	11.97 (5.35)	1.12 (1.58)	11.20 (2.24)
PRM*	3.01 (0.01)	15.60 (2.26)	10.01 (0.01)	14.49 (1.65)
LazyPRM	3.02 (0.00)	12.13 (1.17)	10.02 (0.01)	12.48 (1.96)
RRTstar*	3.01 (0.01)	12.76 (0.93)	10.02 (0.02)	11.47 (1.03)
PRMstar*	3.02 (0.01)	14.43 (1.90)	10.02 (0.01)	12.99 (1.67)
LazyPRMstar	3.02 (0.00)	11.53 (1.23)	10.03 (0.01)	10.95 (1.59)
FMT	2.07 (0.45)	10.49 (0.99)	1.78 (0.21)	10.33 (0.64)
BFMT	1.17 (0.36)	11.74 (2.65)	0.89 (0.09)	10.88 (1.06)
TRRT	0.57 (0.58)	10.21 (0.52)	2.41 (2.82)	10.12 (0.45)
BiTRRT	0.13 (0.08)	15.56 (16.75)	0.13 (0.10)	11.03 (5.22)
SPARS*	3.04 (0.04)	23.63 (4.74)	10.07 (0.07)	23.87 (5.57)
SPARStwo*	3.00 (0.00)	22.98 (3.97)	10.00 (0.00)	26.34 (10.01)

Standard deviation in parentheses

TABLE IV: Average values for benchmark 2

Planner name	Max. 3s computing time		Max. 10s computing time	
	Time (s)	Path length	Time (s)	Path length
SBL	0.05 (0.01)	9.48 (7.86)	0.04 (0.01)	7.76 (1.76)
EST	0.20 (0.13)	9.53 (2.58)	0.16 (0.11)	9.38 (4.00)
BiEST	0.09 (0.04)	11.36 (1.96)	0.08 (0.03)	12.71 (3.57)
ProjEST	0.18 (0.11)	9.86 (2.51)	0.15 (0.09)	8.99 (1.32)
KPIECE	0.18 (0.09)	9.10 (1.50)	0.14 (0.08)	9.49 (1.68)
BKPIECE	0.11 (0.11)	9.71 (5.83)	0.13 (0.16)	8.17 (2.79)
LBKPIECE	0.09 (0.06)	9.33 (5.53)	0.08 (0.03)	9.23 (3.80)
RRT	0.53 (0.62)	12.03 (7.08)	0.44 (1.10)	10.15 (1.99)
RRTConnect	0.09 (0.04)	13.90 (10.39)	0.06 (0.02)	9.65 (4.05)
PDST	0.24 (0.16)	11.71 (3.56)	0.24 (0.16)	12.27 (4.00)
STRIDE	0.19 (0.21)	9.42 (3.26)	0.14 (0.09)	8.98 (1.17)
PRM*	3.02 (0.01)	12.91 (3.41)	10.01 (0.00)	11.56 (1.56)
LazyPRM	3.02 (0.00)	9.80 (1.88)	10.02 (0.00)	9.58 (1.27)
RRTstar	3.01 (0.02)	8.78 (0.00)	10.01 (0.01)	8.21 (0.94)
PRMstar*	3.03 (0.01)	12.30 (1.81)	10.02 (0.01)	11.11 (1.65)
LazyPRMstar	3.02 (0.00)	8.62 (0.98)	10.02 (0.01)	7.88 (0.72)
FMT	1.23 (0.16)	9.64 (6.44)	1.10 (0.15)	7.92 (1.15)
BFMT	0.79 (0.06)	8.24 (0.69)	0.73 (0.06)	8.35 (1.64)
TRRT	0.78 (1.00)	7.82 (0.91)	2.05 (2.43)	8.55 (2.17)
BiTRRT	0.08 (0.02)	8.39 (3.00)	0.07 (0.02)	7.75 (1.11)
SPARS*	3.05 (0.04)	19.38 (8.05)	10.07 (0.06)	16.22 (5.38)
SPARStwo*	3.00 (0.01)	14.98 (5.37)	10.00 (0.00)	20.10 (9.43)

Standard deviation in parentheses

TABLE V: Average values for benchmark 3

Planner name	Max. 3s computing time		Max. 10s computing time	
	Time (s)	Path length	Time (s)	Path length
SBL	1.54 (0.69)	13.72 (5.89)	2.06 (1.56)	15.02 (6.16)
EST	1.06 (0.64)	12.91 (2.14)	1.03 (0.79)	12.84 (1.89)
BiEST	0.11 (0.00)	16.57 (0.00)	3.51 (3.03)	16.99 (4.03)
ProjEST	0.98 (0.74)	13.26 (2.44)	1.21 (1.03)	13.02 (1.87)
KPIECE	1.15 (0.86)	13.31 (3.07)	1.00 (0.90)	13.47 (6.05)
BKPIECE	1.32 (0.78)	21.42 (31.12)	1.19 (0.87)	17.10 (7.24)
LBKPIECE	1.50 (0.88)	14.37 (3.13)	1.34 (1.28)	14.82 (4.58)
RRT*	- (-)	- (-)	3.82 (2.87)	17.20 (2.46)
RRTConnect	0.62 (0.23)	16.13 (13.07)	0.68 (0.28)	14.17 (3.69)
PDST	1.27 (0.71)	13.21 (2.75)	2.07 (1.77)	14.96 (5.60)
STRIDE	0.89 (0.59)	14.76 (4.44)	1.08 (0.82)	13.00 (1.87)
RRTstar*	3.00 (0.00)	18.06 (0.00)	10.01 (0.00)	17.64 (0.00)
FMT	2.91 (0.15)	14.99 (3.77)	6.57 (1.23)	13.47 (0.03)
TRRT	0.90 (0.66)	12.47 (3.10)	1.74 (1.92)	13.07 (4.57)
BiTRRT	1.74 (0.62)	12.55 (2.59)	2.65 (1.63)	16.12 (10.42)

Standard deviation in parentheses

TABLE VI: Average values for benchmark 4

Planner name	Max. 3s computing time		Max. 10s computing time	
	Time (s)	Path length	Time (s)	Path length
SBL	1.71 (0.73)	24.37 (51.46)	2.40 (1.67)	32.01 (83.89)
BKPIECE	1.00 (0.63)	20.15 (30.20)	1.23 (0.80)	20.14 (30.63)
LBKPIECE	1.21 (0.60)	22.68 (35.48)	1.49 (1.10)	21.29 (31.37)
RRTConnect	0.75 (0.32)	21.15 (30.21)	0.91 (0.54)	15.66 (14.27)
PRM*	- (-)	- (-)	10.01 (0.00)	18.22 (0.00)
BiTRRT	1.61 (0.65)	28.14 (35.93)	3.05 (2.02)	28.88 (62.82)

Standard deviation in parentheses

Computing time. When feasible paths need to be found quickly, a bi-directional planner is recommended similar motion planning problems to benchmark 1, 2 and 4. When a path creation starts in a narrow passage as in benchmark 3, a single-directional planner can give improved computing times, with exception of RRTConnect.

Path length. Picking an optimizing planner to find shorter path lengths can not be justified from the benchmark data. Only in benchmark 2, the TRRT and BiTRRT planners for 3s and 10s maximum computing time respectively compute shorter paths compared to non-optimizing planners. Having more computing time for time-variant goals decreases path lengths, however, not drastically. The RRTConnect shows low path length means in all benchmark problems.

Multi-query. For this paper, we only looked at single-query motion planning problems, multi-query planners can also be used as single-query planners. Though this paper fails to show the potential benefit of using the same roadmap multiple times and how this can affect computing time. Though we do notice that RRTConnect and BiTRRT are able to give valid paths in very short amounts of time that we argue the need for multi-query planners for online grasp executions.

Parameter selection. Since parameter values have to be set for a planner to operate, the aim was to achieve maximum performance of the planners. By manually conducting the iterative process explained before, a guarantee of maximum performance can not be given. We executed the iterative process to the best of our ability to achieve maximum performance.

Combined metrics. When looking at all the benchmarks, SBL, BKPIECE, LBKPIECE, RRTConnect and BiTRRT show high performance in the metrics solved runs, computing time and path length.

Optimization with a time-invariant goal. Planners FMT, BFMT, LBTRRT, TRRT and BiTRRT stop once an optimized path is found. Of these, BiTRRT is the fastest performing planner. However, compared to not non-optimizing planners the path length is not consistently shorter. More research has to be done to see whether other metrics will make the use of this planner more desirable.

Robot. The UR5 is used to compute the motion plans though we have not investigated the change in the performance measures for different types of manipulators. It needs to be determined if the results hold for other types of manipulators. For future work, multiple manipulators can be used to perform the same benchmark to find a better answer on the consistency of the planner's performance.

Path constraints. The results presented do not give any estimation on how the planners perform when implementing hard path constraints.

VII. Conclusion

This paper presented benchmark data of available OMPL planners in MoveIt! for geometrically constrained grasp executions using a 6-DOF manipulator. Planner performance was studied by means of solved runs, computing time and path length for two maximum computing time settings. From the performance analysis remarks and recommendations were made depending on the performance measure. For the defined benchmarks, the bi-directional planners SBL, LBKPIECE, RRTConnect and BiTRRT are high performing planners in terms of all the studied metrics. For future work, we would like to investigate the use of different manipulators to find consistency in planner performance when performing grasp executions.

References

[1] I. A. Sucan and S. Chitta, "MoveIt!" [Online]. Available: http://moveit.ros.org

[2] I. Sucan, M. Moll, and L. Kavraki, "Open Motion Planning Library: A Primer," 2014.

[3] M. Moll and A. S. Ioan, "Benchmarking Motion Planning Algorithms," *Ieee Robotics & Automation Magazine*, no. August, 2015.

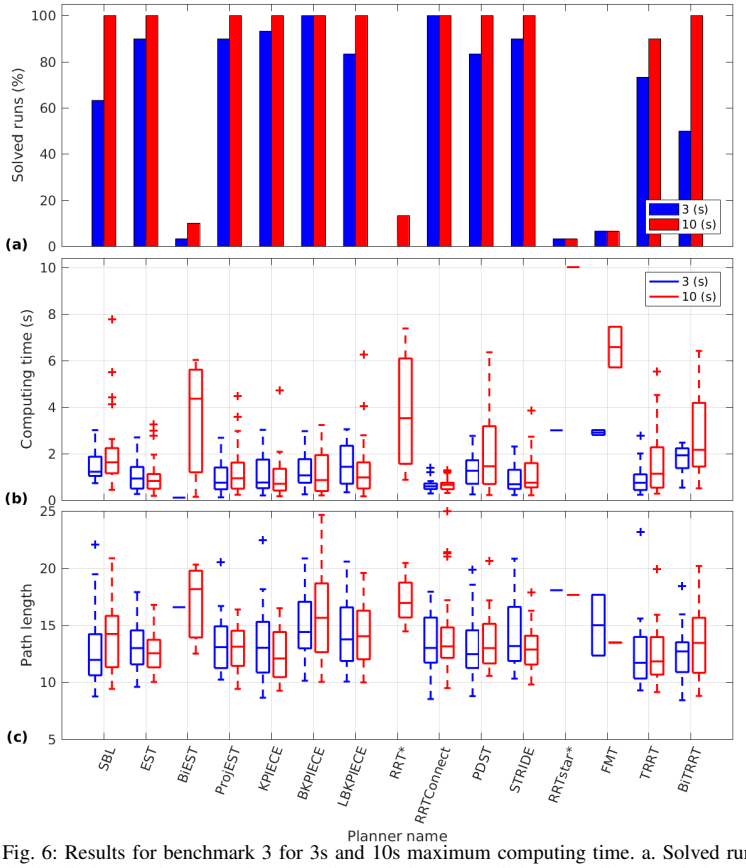


Fig. 6: Results for benchmark 3 for 3s and 10s maximum computing time. a. Solved runs (higher is better), b. Computing time (lower is better, small interquartile range is better), c. Path length (lower is better, small interquartile range is better)

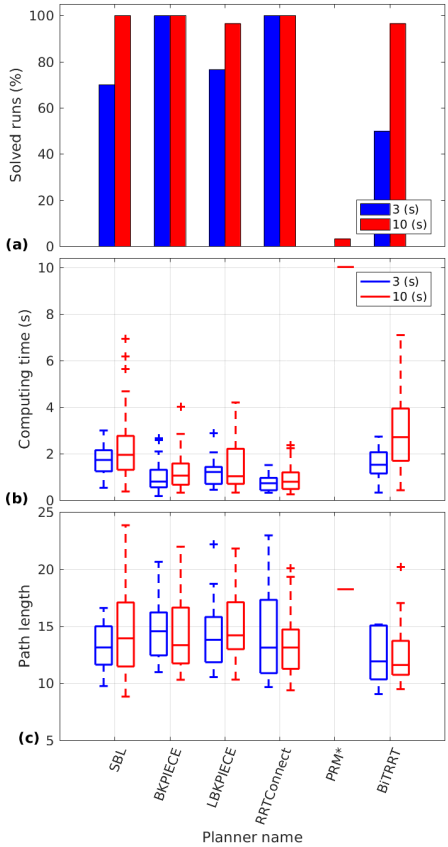


Fig. 7: Results for benchmark 4 for 3s and 10s maximum computing time. a. Solved runs (higher is better), b. Computing time (lower is better, small interquartile range is better), c. Path length (lower is better, small interquartile range is better)

[4] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," 1998.

[5] G. Sánchez and J.-C. Latombe, "A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking," in *Robotics Research*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 403–417.

[6] D. Hsu, J. C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of International Conference on Robotics and Automation*, vol. 3, 1997, pp. 2719–2726 vol.3.

[7] I. A. Sucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *IN WORKSHOP ON THE ALGORITHMIC FOUNDATIONS OF ROBOTICS*, 2008.

[8] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *In IEEE Int. Conf. Robot. Autom*, 2000, pp. 521–528.

[9] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[10] J. J. K. Jr. and S. M. Lavalle, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. IEEE Intl Conf. on Robotics and Automation*, 2000, pp. 995–1001.

[11] A. M. Ladd, R. Unversity, L. E. Kavraki, and R. Unversity, "Motion planning in the presence of drift, underactuation and discrete system changes," in *In Robotics: Science and Systems I*. MIT Press, 2005, pp. 233–241.

[12] B. Gipson, M. Moll, and L. E. Kavraki, "Resolution Independent Density Estimation for motion planning in high-dimensional spaces," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, may 2013, pp. 2437–2443.

[13] L. Kavraki, P. Svestka, J. claude Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, 1996, pp. 566–580.

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

[15] L. Janson and M. Pavone, "Fast marching trees: a fast marching sampling-based method for optimal motion planning in many dimensions - extended version," *CoRR*, vol. abs/1306.3532, 2013.

[16] J. A. Starek, J. V. Gómez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," *CoRR*, vol. abs/1507.07602, 2015.

[17] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality, motion planning," *CoRR*, vol. abs/1308.0189, 2013. [Online]. Available: http://arxiv.org/abs/1308.0189

[18] L. Jaillet, J. Corts, and T. Simon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, pp. 635–646, 2010.

[19] D. Devaurs, T. Siméon, and J. Cortés, "Enhancing the transition-based rrt to deal with complex cost spaces," in *IEEE International Conference on Robotics and Automation, ICRA '13*, Karlsruhe, Germany, 2013, pp. 4105–4110.

[20] A. Dobson, A. Krontiris, and K. E. Bekris, *Sparse Roadmap Spanners*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 279–296.

[21] A. Dobson and K. E. Bekris, "Improving sparse roadmap spanners," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.