Performance study of single-query motion planning for grasp execution using various manipulators

Meijer, Jonathan; Lei, Qujiang; Wisse, Martijn

Publication date
2017

Document Version
Accepted author manuscript

Published in
Proceedings of the18th International Conference on Advanced Robotics

Performance Study of Single-Query Motion Planning for Grasp Execution Using Various Manipulators

Jonathan Meijer
TU Delft Robotics Institute
Delft University of Technology
Delft, The Netherlands
j.g.j.meijer@student.tudelft.nl

Qujiang Lei
TU Delft Robotics Institute
Delft University of Technology
Delft, The Netherlands
q.lei@tudelft.nl

Martijn Wisse
TU Delft Robotics Institute
Delft University of Technology
Delft, The Netherlands
m.wisse@tudelft.nl

Abstract— **This paper identifies high performing motion planners among three manipulators when carrying out grasp executions. Simultaneously, this paper presents useful benchmarking data. Sampling-based motion planners of OMPL available for use in MoveIt! are compared by performing several grasping-related motion planning problems. The performance of the planners is 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 for the used manipulators.**

I. Introduction

Currently, 23 sampling-based motion planners of OMPL (Open Motion Planning Library) [1] are available for use in MoveIt! [2], a robot manipulation framework for ROS. OMPL is the default and most supported motion planning library in MoveIt!. Support for picking a planner is not provided. In the literature, no research about planner performance can be found when performing grasp executions. Moreover, performance information for 12 planners is scarce, since they have just been released (Decemeber 2016). This leaves the user to perform time-consuming benchmarks in order to find the right planner.

This paper aims to identify high performing OMPL motion planners available in MoveIt! when carrying out grasp executions. Simultaneously, it aims to present useful benchmarking data for all the 23 planners. By conducting several grasp executions for three different manipulators, we hope to find planners that consistently show high performance. We will use three manipulators that have different geometry but have similar specifications. These are Universal Robots UR5, KUKA LWR 4+ and Kinova JACO. To resemble a real-world grasping problem, the manipulators are fitted with a gripper. In this study, we only consider sampling-based motion planners of OMPL available in MoveIt!, listed in Tab. I. This includes so-called multi-query planning methods. However, only single-query performance is measured. We have designed a virtual environment that resembles a shelf in which objects can be picked or placed. Planner choice is investigated by considering geometry constraints for two motion planning problems. For the third problem we add a path constraint.

Due to randomization of sampling-based motion planners, motion planning problems have to be run multiple times to provide saturated results on the performance. The performance of the planners is measured in terms of solved runs, computing time and path length. The metric solved runs can be expressed as a percentage of the total motion planning runs that finish correctly. Barplots are used to visualize the difference. For every run, 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 need of the user, high performance for one metric can result in low performance for an other metric. By analyzing each metric separately, we can elaborate on right planners for each metric.

Tab. I shows there are optimizing OMPL planners available in MoveIt! that have a time-invariant goal. This means they stop computing as soon as a path is found that is considered to be more optimal. However, compared to non-optimizing planners, computing effort is increased which can result in an increase in computing time. For optimizing planners is expected that they will produce shorter path lengths. This study can clarify if extra computing time of optimizing planners will considerably decrease the path length compared to non-optimizing planners.

II. Background

A. Manipulators

Universal Robots UR5 [3]: Universal Robots aims to provide easily programmable, safe and flexible industrial robots. The UR5 manipulator is designed to be lightweight, flexible and collaborative. The manipulator has 6 degrees of freedom (DOF). Joint limits are max 2π in both directions. The manipulator has a maximum payload of 5kg and a span of 850mm. A model of the UR5 fitted with a gripper is shown in Fig. 1.

KUKA LWR 4+ [4]: KUKA supplies intelligent automation solutions and is currently one of the top brands in this field. The 7-DOF LWR 4+ is a lightweight collaborative manipulator. Its furthest reach is 1178.5mm and it can lift up to 7kg. The manipulator is made for universal purpose, meaning it can be used for many applications. The upper and lower limits are ±170 degrees for four joints and ±120 degrees for the remaining joints. A model of the LWR 4+ is shown in Fig. 2.

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

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

Planner name & Reference		Optimizing planners	Multi-query	Time-invariant goal
SBL	[6]			✓
EST	Based on [7]			✓
BiEST	[7]			✓
ProjEST	Based on [7]			✓
KPIECE	[8]			✓
BKPIECE	Based on [8]			✓
LBKPIECE	Based on [8][9]			✓
RRT	[10]			✓
RRTConnect	[11]			✓
PDST	[12]			✓
STRIDE	[13]			✓
PRM	[14]		✓	
LazyPRM	[9]		✓	
RRTstar	[15]	✓		
PRMstar	Based on [14][15]	✓	✓	
LazyPRMstar	Based on [9][15]	✓	✓	
FMT	[16]	✓		✓
BFMT	[17]	✓		✓
LBTRRT	[18]	✓		✓
TRRT	[19]	✓		✓
BiTRRT	[20]	✓		✓
SPARS	[21]	✓	✓	
SPARStwo	[22]	✓	✓	

Kinova JACO^1 and JACO^2 [5]: Kinova has developed two JACO versions, the JACO^1 and the JACO^2. They are lightweight and geometrically identical. The manipulators have a maximum reach of 900mm. Other than the UR5 and LWR 4+, which have straight links, this manipulator has two links which have a curve. Both versions have three joints that can rotate continuously, remaining joints have limits due to the geometry. The maximum payload is 1.5kg for the JACO^1 and 2.6kg for the JACO^2. The manipulator has 6 degrees of freedom. A model is shown in Fig. 3.

B. Software

The open-source Robot Operating System (ROS) is a suite of software libraries and was created to encourage collaborative robotics software development. Inside ROS, the MoveIt! framework deals with the manipulation of robotic hardware. Several motion planner libraries can be configured with MoveIt! to help solve a motion planning problem. OMPL (Open Motion Planning Library) is the most supported motion planner library and is the default library for MoveIt!. The library consists of state of the art sampling-based motion planners.

Using 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.

C. Overview of planners

Sampling-based motion planners are proven to be probabilistic complete [14], which implicates that the probability of not finding a feasible path in an unbounded setting approaches zero. Therefore, these planners are widely used to find feasible paths in high-dimensional and geometrically constraint environments. Optimizing sampling-based motion planners can refrain from potential high-cost paths and rough motions [15]. However, computing effort for finding an optimized path is increased.

One of the most common is sampling-based motion planner is the Probabilistic RoadMap (PRM) [14]. The planner makes a roadmap by sampling random states in the configuration space and mark them as nodes (vertices). Nodes are connected to other nearby nodes if this path segment (edge) is collision-free. Once the construction of the roadmap is finished a graph search is performed in order to find a connection from the initial state towards the goal state. The roadmap of the PRM planner attempts to cover the total free configuration space, making it suitable to reuse the roadmap for a different motion planning problem in the same configuration space. This is referred to as a multi-query planning method. Among the 23 motion planners in Tab. I, six can be considered as multi-query planning methods.

The LazyPRM [9] planner is different from the PRM method, since it initially accepts invalid configuration states to construct a roadmap. After the graph search is performed, the invalid parts of the candidate solution are altered to make the path collision-free. PRMstar [15] is the asymptotically optimal variant of the PRM planner. It rewires nodes to other near nodes if this minimizes cost towards the node. LazyPRMstar [15] combines the LazyPRM and PRMstar.

The remaining planners in Tab. I are refered to as single-query planning methods. These create a roadmap every time a new planning query has to be solved. A common single-query planner is the Rapidly-exploring Random Tree (RRT) method [10]. It grows a tree structure 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 configuration 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 RRTConnect method [11] 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, which solves the the motion planning problem.

The RRT with an optimizing step is the RRTstar [15] planner. This variant of RRT checks whether the new sampled node can be connected to other near nodes so that the state space is more locally refined, similar to PRMstar. The Lower Bound Tree-RRT (LBT-RRT) [18] planner is another optimizing planner. It 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 [19] 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 Bi-TRRT [20] is a bi-directional version of this planner.

The EST method [7] stands for Expansive Space Trees. The planner tries to determine the direction of the tree by checking the density of nodes in the configuration space.

The tree will expand towards the less explored space. Bi-directional EST (BiEST) [7] grows two trees, similar to RRTConnect. Projection EST (ProjEST), based on [7], detects the less explored area of the configuration space by using a grid. This grid serves as a projection of the configuration space. Single-query Bi-directional probabilistic roadmap planner with Lazy collision checking, also called SBL [6], grows two trees, which expand in the same manner as EST. The planner differentiates from EST by the lazy collision-checking.

KPIECE (Kinodynamic motion Planning by Interior-Exterior Cell Exploration) [8] is a tree-based planner that uses layers of discretization to help estimate the coverage of the state space. There also exists a bi-directional variant called BKPIECE and a variant which incorporates lazy collision checking, this is the LBKPIECE.

Fast Marching Tree (FMT) [16] is an asymptotically optimal planner which marches a tree forward in the cost-to-come space on a specified amount of samples. The BFMT [17] planner is a bi-directional variant of this planner.

PDST (Path-Directed Subdivision Tree) [12] 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) [13] uses a Geometric Nearneighbor Access Tree (GNAT) to estimate the density of the configuration space. This helps to guide the tree 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 [15]. 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:

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

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:

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

For the implementation of motion constraints, the free configuration space is reduced. Only configurations that satisfy the motion constraint can be valid configurations. This can result increase the amount of narrow passages, which are known to cause issues for sampling-based motion planners [23].

Problem implementation. Grasp executions will be simulated in geometry constrained scenes inside the MoveIt! framework to retrieve data on planner performance. Non-optimizing planners are instructed to produce feasible paths. The optimizing planners are instructed to produce optimized paths. The motion constraint problem is defined to keep the gripper horizontal. Rotation of the gripper in the horizontal plane is allowed (max 2π), other rotations are limited to 0.1rad. Path simplification by OMPL for all the motion planning problems is turned on. Multi-query planners are being used as single-query planners.

Performance metric. Solved runs, computing time and path length are used as metrics in our experiments. We analyze the metrics outcome individually to provide the best performing planners in each of the metrics. Solved runs is expressed in terms of the percentage of total runs resulting in feasible or optimized paths. High solved runs is considered as high performance. Computing time is measured for the time it takes for planners to produce feasible paths or optimized paths. Planners with a low computing time are considered as high performance. Path length is measured by the length of the sum of motions for a produced path. Planners with short path length are considered as high performance. Mean and standard deviation values of computing time and path length can provide extra information on the performance. Low mean and small standard deviations values are considered as high performance.

Parameters. For 20 of the 23 OMPL planners in MoveIt!, parameters have to be set in order for the planner to solve a motion planning problem. Choosing right parameter values can improve the performance of the planner. To aim for maximum performance an extensive parameter selection was conducted, since no automatic optimization process is available to this date.

IV. Defined motion planning problems

Three grasp execution motions have been defined to measure the performance of the planners. They are defined in the same environment that consists of a simplified shelf and obstacles.

A. Benchmark 1: Place grasp

Benchmark 1 initial end-effector position is located at the end of a shelf, shown as the orange colored robot state in Fig. 1,2,3. The goal position is a stretched arm configuration in front of the shelf (gray robot state). The motion planning problem starts a in a narrow passage, the goal is situated in a less constrained space. This would identify which planner is able to produce the best results when moving out of a constrained space.

B. Benchmark 2: Pick grasp

Benchmark 2 is attempting to resemble a picking motion from a narrow shelf, shown as the gray robot state in Fig. 1,2,3. The goal of the problem is to achieve a specific gripper orientation at the end of this shelf (orange robot state). The planner will have to produce a motion plan with high accuracy to reach the end of the shelf. The motion

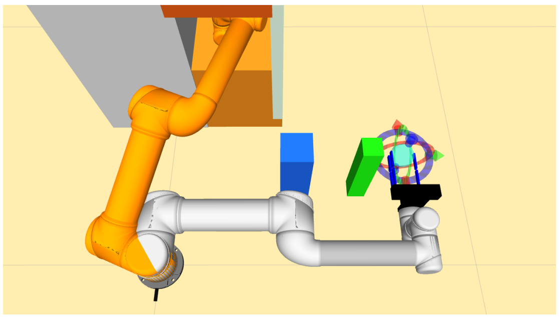
Fig. 1. Pick and place benchmark for UR5.

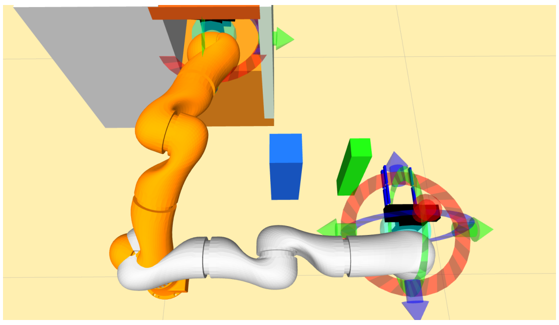
Fig. 2. Pick and place benchmark for KUKA LWR 4+.

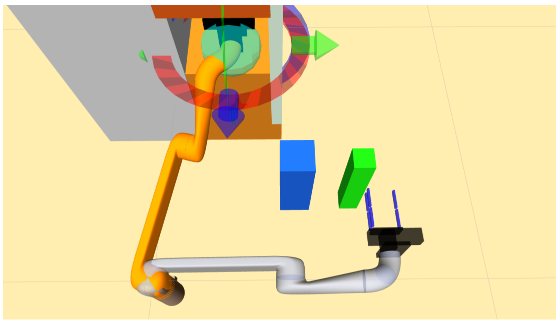
Fig. 3. Pick and place benchmark for Kinova JACO.

planning problem starts in a less constrained space, the goal is situated in a narrow passage. This would identify which planner is able to produce the best results when moving into a constrained space.

C. Benchmark 3: Place grasp with motion constraints

For benchmark 3, the same motion planning problem as benchmark 1 is defined. However, for this problem motion constraints are added to keep the gripper horizontally leveled within a small margin. This resembles placing a glass of water from the shelf on the table, without spilling. This would identify which planner is able to produce the best results when having a constrained free configuration space.

V. 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. Parameter estimation for the planners is conducted using an extensive iterative process to achieve maximum performance of the planners with respect to the manipulator, these planner specific parameters are presented in Tab. II. The global OMPL parameter $longest\_valid\_segment\_fraction$ is set to 0.005. To give reliable data on the solved runs, computing time and path length, each algorithm was run 50 times for the given motion planning problem. The planners were given a maximum computing time 10s for benchmarks 1 and 2. Due to the increased limitation of the free configuration space of motion constraint planning, benchmark 3 was given 20s maximum computing time. The time is kept low since most robotics applications need to operate quickly.

B. Simulation results

Results of benchmark 1 are shown in Fig. 4 and Tab. III. Considering all manipulators, solved runs of 80% and higher were found for all single-query planners, except for FMT. Since multi-query planners do not focus on one specific motion planning problem, the roadmap construction needs to cover the whole $\mathcal{C}_{\text{free}}$. A demerit of this is the extra needed computing effort. Single-query planners do not need to cover the total free configuration space. Heuristics of these planners help propagating a path outwards of a constrained space. Lowest computing times were retrieved with EST, ProjEST, KPIECE and STRIDE, which are all mono-directional planners with a *goal bias* property. Since the goal configuration is located in a large free space, the probability of finding a solution with the goal configuration as sample increases. The fastest planners also use heuristics to quickly cover the configuration space. EST and STRIDE do this by looking at the density of present samples. KPIECE uses a discretization layer which coarsely covers the configuration space. With goal bias in an open space, short paths can be found with EST, ProjEST and KPIECE.

When considering the results of manipulators in benchmark 1 separately, it can be noted that the JACO manipulator was able to generate higher solved runs for multi-query planners. This manipulator does not incorporate any restrictions on joint limits, which helps to find more connections in the free configuration space between nodes, increasing the solved runs. Moreover, a solution faster is found faster and with a shorter path length. Computing times for the UR5 manipulator were lowest with KPIECE, RRT and RRTConnect. Computing times for the LWR 4+ were lower than the UR5 with SBL, EST, KPIECE, BKPIECE, LBKPIECE and STRIDE. In addition to those planners, RRTConnect also had low computing times for the JACO. For the UR5, short path lengths are found with BiEST, ProjEST, KPIECE, RRTConnect, TRRT and BiTRRT. For the LWR 4+, these are EST, KPIECE, BKPIECE, LBKPIECE and LazyPRMstar. Planners EST, BiEST, ProjEST, KPIECE, LazyPRMstar and BiTRRT found short paths for the JACO manipulator.

Results of benchmark 2 are shown in Fig. 5 and Tab. IV. Considering all manipulators, solved runs of 80% and higher were retrieved with SBL, BKPIECE, LBKPIECE, RRTConnect and BiTRRT. These are all bi-directional tree-based planners. Because of this property path planning is

TABLE II: PLANNER PARAMETERS FOR UR5 (U), LWR 4+ (L) AND JACO (J)

SBL	U	L	J	EST	U	L	J	BiEST	U	L	J	ProjEST	U	L	J	RRT	U	L	J	RRTConnect	U	L	J
range	0.525	0.6	0.6	range	0.6	0.6	0.6	range	0.6	0.6	0.6	range	0.45	0.6	0.6	range	0.75	1.2	1.8	range	0.2	0.6	0.6
				goal bias	0.05	0.05	0.075					goal bias	0.075	0.025	0.075	goal bias	0.075	0.075	0.025				

PRM	U	L	J	LazyPRM	U	L	J	RRTstar	U	L	J	KPIECE	U	L	J	BKPIECE	U	L	J	LBKPIECE	U	L	J
max n.n.	10	10	10	range	0.525	0.6	0.6	range	0.75	1.2	0.6	range	0.525	0.3	0.225	range	0.525	0.3	0.225	range	0.6	0.225	0.3
								goal bias	0.075	0.05	0.05	goal bias	0.075	0.05	0.075	border_f.	0.9	0.9	0.8	border_f.	0.9	0.8	0.8
								delay c.c.	1	1	1	border_f.	0.9	0.9	0.8	exp. s.f.	0.7	0.5	0.5	valid p.f.	0.5	1	0.5
												exp. s.f.	0.7	0.5	0.5	valid p.f.	0.5	0.5	0.5				
												valid p.f.	0.5	0.5	0.5								

STRIDE	U	L	J	FMT	U	L	J	TRRT	U	L	J	BiTRRT	U	L	J	SPARS	U	L	J	SPARStwo	U	L	J
range	0.6	0.45	0.45	num samp.	1000	1000	1000	range	1.35	1.2	1.8	range	0.6	0.9	0.6	stretch f.	3	2.6	2.6	stretch f.	3	3	3
goal bias	0.05	0.05	0.075	rad. Mult.	1.15	1.15	1.15	goal bias	0.075	0.05	0.025	temp c.f.	0.3	0.3	0.3	sparse d.f.	0.25	0.25	0.25	sparse d.f.	0.25	0.25	0.25
use proj d.	0	0	0	Nearest k	1	1	1	max state f.	5	5	5	init temp	75	75	75	dense d.f.	0.001	0.001	0.001	dense d.f.	0.001	0.001	0.001
degree	24	8	8	cache cc	1	1	1	temp c.f.	2.5	2	2	frountier	1	1	1	max fail.	1000	1000	1000	max fail.	5000	5000	5000
max deg.	28	12	12	heuristics	1	1	1	min temp	1e-9	1e-9	1e-9	frountierN.r.	0.1	0.1	0.1								
min deg.	12	6	6	ext. fmt	1	1	1	init temp	125	125	125	cost thres.	1000	1000	1000								
max p.p.l.	6	3	3					frountier	2.25	2.5	2.5												
est. dim.	0	0	0					frountierN.r.	0.1	0.1	0.1												
valid p.f.	0.1	0.1	0.1					k constant	0	0	0												

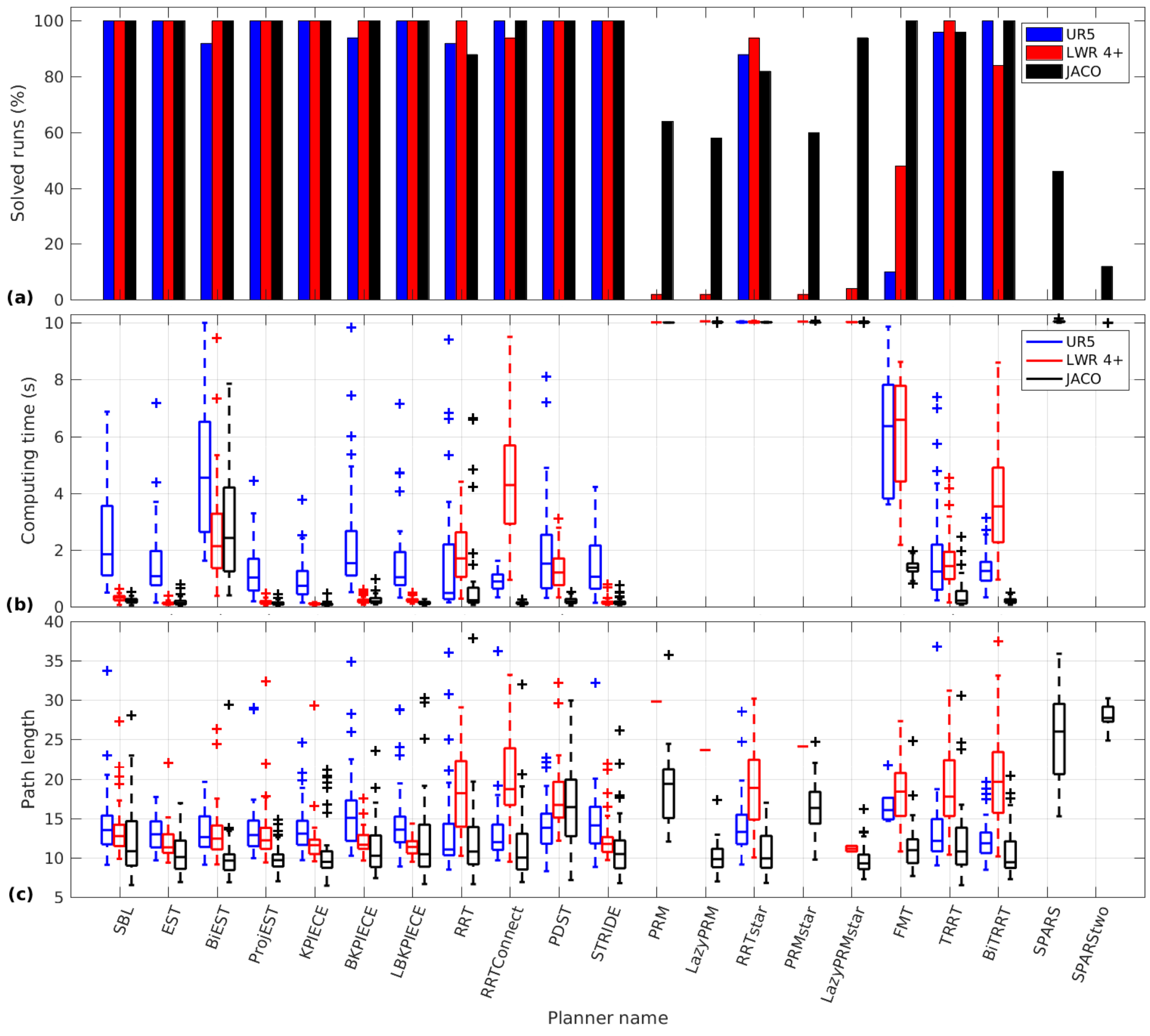

Fig. 4. Results for benchmark 1. (a) Solved runs; higher is better. (b) Computing time; lower is better, small interquartile range is better. (c) Path length; lower is better, small interquartile range is better.

also started in the goal state, which acts similar to the benchmark 1 moving out of a constrained space. SBL and LBKPIECE showed the lowest computing times. These planners use lazy collision-checking. Collision-checking is only performed on a candidate path instead on all vertices, which can lower the computing time. Shortest paths are found with SBL and LBKPIECE, however, standard deviations are higher for the SBL planner. Using a discretization layer to cover the configuration space coarsely helps finding more consistent results (lower standard deviations from the mean).

When considering the results of manipulators in benchmark 2 separately, it can be noted that the UR5 and LWR 4+ also managed to produce solved runs of 80% and

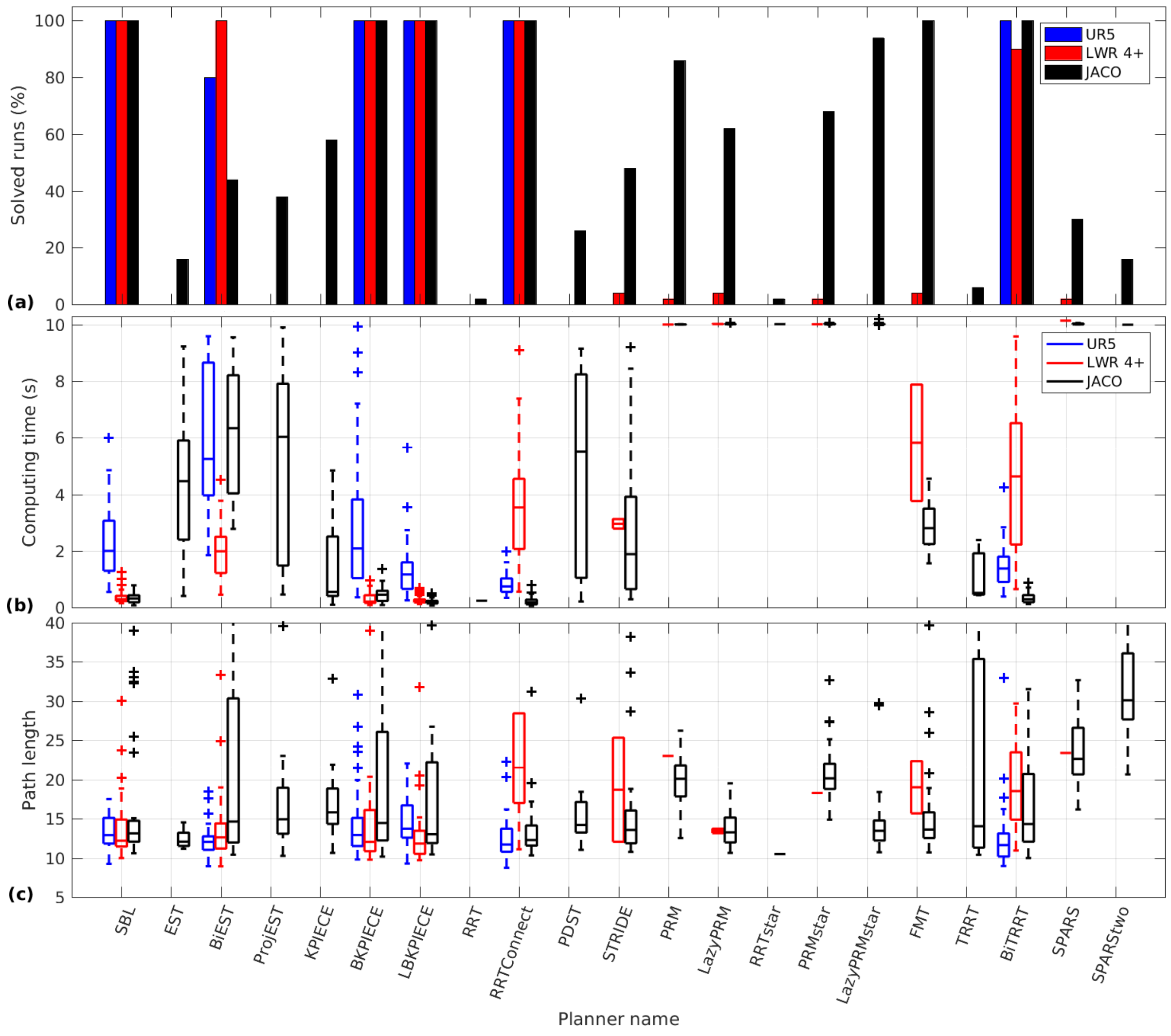

Fig. 5. Results for benchmark 2. (a) Solved runs; higher is better. (b) Computing time; lower is better, small interquartile range is better. (c) Path length; lower is better, small interquartile range is better.

higher for the BiEST planner. This planner is also a bi-directional planner. The expanded configuration space of the JACO manipulator helps to produce solved runs of 80% and higher for planners PRM, PRMstar and LazyPRMstar. However, computing times are considerably higher since these planners keep optimizing the roadmap until the time-limit is reached. RRTConnect is the fastest planner for the UR5 and LBKPIECE is the fastest for LWR 4+ and JACO. For all three manipulators, LBKPIECE is able to compute feasible paths within 1.5s. Considering high solved runs: BiTRRT finds the shortest paths for the UR5, SBL for the LWR 4+ and FMT for the JACO respectively.

Results of benchmark 3, incorporating motion constraints, are shown in Fig. 6 and Tab. V. The extra constraint limits the free configuration space. Considering all manipulators, only the BiEST planner was able to produce solved runs of 80% and higher. The planner looks at the density of samples in its neighbourhood to help its expansion. In the case of planning with motion constraints, this shows to be effective compared to the other 20 planners. The bi-directional property helps finding a solution within the time limit, since the mono-directional EST planner is not able to find a feasible path with a maximum computing time of 20s.

Depending on the manipulator, the highest performing planner differs, which indicates that there is less consistency in planner performance when incorporating extra motion constraints. SBL, BKPIECE and RRTConnect also had solved runs of 80% and higher for the UR5 manipulator. For the JACO manipulator the KPIECE planner manages to get solved runs of 100%.

C. Discussion

The motivation of this work is to help users pick high-performing motion planners for grasp executions. Shortcomings of this work will be discussed.

Computing time. This paper only showed results for motion planning with a time-constraint of 10s or 20s. This was chosen to select high-performing motion planners that find a solution in a timely manner. Selecting a higher time-limit could show increased performance in solved runs and path length. However, this is not covered in this work.

TABLE III: MEAN VALUES FOR BENCHMARK 1

Planner name	UR5		LWR 4+		JACO	
	Time (s)	Path length	Time (s)	Path length	Time (s)	Path length
SBL	2.42 (1.68)	14.24 (4.11)	0.31 (0.11)	13.38 (3.23)	0.21 (0.09)	15.93 (19.85)
EST	1.54 (1.26)	14.57 (7.50)	0.12 (0.06)	11.82 (2.00)	0.16 (0.15)	10.55 (2.41)
BiEST	4.71 (2.25)	13.73 (4.93)	2.55 (1.77)	13.02 (3.21)	2.98 (2.04)	10.63 (5.47)
ProjEST	1.22 (0.89)	13.70 (3.73)	0.15 (0.08)	14.31 (9.77)	0.12 (0.07)	10.71 (5.77)
KPIECE	0.97 (0.73)	13.72 (3.07)	0.10 (0.03)	11.91 (2.86)	0.09 (0.06)	10.73 (3.62)
BKPIECE	2.31 (1.94)	16.55 (7.70)	0.21 (0.11)	11.92 (1.47)	0.23 (0.16)	11.22 (3.39)
LBKPIECE	1.51 (1.30)	14.45 (4.16)	0.23 (0.09)	11.48 (1.08)	0.13 (0.05)	12.17 (5.20)
RRT	1.59 (2.04)	14.39 (9.07)	1.94 (1.12)	18.68 (6.05)	0.83 (1.59)	12.14 (5.10)
RRTConnect	0.89 (0.35)	13.06 (3.98)	4.49 (2.06)	21.34 (10.20)	0.13 (0.06)	13.88 (12.11)
PDST	1.88 (1.62)	16.27 (17.34)	1.25 (0.61)	25.74 (36.23)	0.20 (0.11)	16.18 (5.22)
STRIDE	1.46 (1.13)	16.36 (15.01)	0.16 (0.14)	12.35 (2.51)	0.16 (0.13)	14.66 (20.43)
PRM			10.02 (0.00)	29.82 (0.00)	10.01 (0.00)	18.73 (4.55)
LazyPRM			10.05 (0.00)	23.66 (0.00)	10.02 (0.01)	10.11 (2.17)
RRTstar	10.02 (0.01)	14.32 (4.15)	10.03 (0.02)	19.06 (5.04)	10.02 (0.01)	10.66 (2.73)
PRMstar			10.04 (0.00)	24.11 (0.00)	10.02 (0.01)	16.65 (3.28)
LazyPRMstar			10.02 (0.00)	11.13 (0.50)	10.02 (0.01)	9.79 (1.86)
FMT	6.17 (2.57)	16.71 (2.88)	6.06 (1.92)	18.29 (4.32)	1.39 (0.25)	11.30 (2.98)
TRRT	1.76 (1.68)	13.40 (4.31)	1.63 (0.93)	20.47 (12.40)	0.41 (0.51)	13.48 (8.55)
BiTRRT	1.31 (0.60)	13.79 (11.02)	3.61 (1.84)	21.05 (8.62)	0.20 (0.10)	10.77 (3.19)
SPARS					10.04 (0.03)	25.52 (6.04)
SPARStwo					10.00 (0.00)	27.83 (1.84)

Standard deviation in parentheses
Gray cells $\rightarrow$ *time* within $2 \cdot \text{time}_{\text{min}}$ and *path length* within $1.2 \cdot \text{path\_length}_{\text{min}}$, for solved runs $> 80\%$

TABLE IV: MEAN VALUES FOR BENCHMARK 2

Planner name	UR5		LWR 4+		JACO	
	Time (s)	Path length	Time (s)	Path length	Time (s)	Path length
SBL	2.30 (1.36)	35.21 (99.24)	0.36 (0.22)	20.66 (37.57)	0.33 (0.16)	19.85 (18.23)
EST					4.40 (2.94)	12.41 (1.20)
BiEST	5.82 (2.52)	20.99 (39.61)	1.99 (0.95)	26.90 (35.64)	6.37 (2.26)	24.80 (20.19)
ProjEST					5.13 (3.25)	18.57 (10.76)
KPIECE					1.37 (1.38)	22.36 (20.44)
BKPIECE	2.83 (2.34)	16.55 (17.33)	0.29 (0.21)	37.36 (73.78)	0.45 (0.25)	24.45 (20.39)
LBKPIECE	1.34 (0.95)	19.01 (23.57)	0.28 (0.15)	21.24 (31.19)	0.20 (0.08)	21.46 (18.29)
RRT					0.24 (0.00)	61.77 (0.00)
RRTConnect	0.81 (0.36)	26.84 (63.29)	3.62 (1.89)	72.72 (112.46)	0.21 (0.14)	20.70 (30.91)
PDST					5.02 (3.42)	15.83 (4.94)
STRIDE			2.96 (0.24)	18.69 (9.38)	2.75 (2.54)	17.40 (10.26)
PRM			10.01 (0.00)	23.00 (0.00)	10.01 (0.00)	19.93 (2.99)
LazyPRM			10.03 (0.00)	13.43 (0.40)	10.02 (0.01)	13.97 (2.58)
RRTstar					10.02 (0.00)	10.48 (0.00)
PRMstar			10.02 (0.00)	18.28 (0.00)	10.03 (0.02)	20.68 (3.62)
LazyPRMstar					10.03 (0.05)	14.29 (3.88)
FMT			5.82 (2.92)	19.01 (4.71)	2.87 (0.83)	17.15 (12.78)
TRRT					1.11 (1.10)	22.31 (17.58)
BiTRRT	1.45 (0.70)	14.49 (13.19)	4.47 (2.57)	65.18 (125.80)	0.33 (0.17)	25.30 (23.12)
SPARS			10.15 (0.00)	23.39 (0.00)	10.03 (0.02)	23.24 (4.76)
SPARStwo					10.00 (0.00)	31.47 (7.23)

Standard deviation in parentheses
Gray cells $\rightarrow$ *time* within $2 \cdot \text{time}_{\text{min}}$ and *path length* within $1.2 \cdot \text{path\_length}_{\text{min}}$, for solved runs $> 80\%$

TABLE V: MEAN VALUES FOR BENCHMARK 3

Planner name	UR5		LWR 4+		JACO	
	Time (s)	Path length	Time (s)	Path length	Time (s)	Path length
SBL	8.88 (4.19)	20.66 (8.09)			12.44 (0.00)	13.55 (0.00)
EST	8.99 (7.49)	18.03 (4.03)	9.08 (0.00)	67.08 (0.00)	11.06 (4.46)	12.25 (1.48)
BiEST	2.81 (1.21)	21.18 (10.97)	8.77 (4.36)	35.71 (26.06)	8.72 (6.00)	27.84 (16.28)
ProjEST					10.06 (5.40)	13.41 (1.25)
KPIECE	9.08 (3.45)	26.58 (12.90)	9.50 (6.58)	41.99 (39.29)	3.84 (2.70)	15.10 (4.43)
BKPIECE	5.72 (4.82)	24.92 (13.76)	9.92 (4.51)	40.82 (30.96)	9.78 (7.12)	44.91 (24.50)
LBKPIECE	15.21 (4.39)	25.39 (13.22)			9.88 (0.00)	12.85 (0.00)
RRT	9.06 (5.63)	29.45 (24.11)	12.32 (5.71)	43.74 (42.21)	12.10 (5.65)	12.28 (1.80)
RRTConnect	4.86 (3.63)	27.92 (23.82)	13.03 (4.00)	65.97 (55.08)	10.74 (5.79)	26.76 (16.25)
PDST	9.23 (0.00)	11.77 (0.00)	12.42 (5.30)	41.06 (39.27)	8.86 (6.06)	51.48 (32.37)
STRIDE	9.98 (10.20)	58.23 (60.47)	12.64 (0.00)	13.15 (0.00)	12.45 (7.28)	23.87 (15.41)
PRM	20.12 (0.04)	65.88 (12.16)	20.14 (0.05)	43.60 (7.56)	20.09 (0.05)	55.03 (20.03)
RRTstar	20.07 (0.04)	22.89 (13.78)	20.03 (0.01)	25.18 (11.74)	20.05 (0.03)	14.97 (3.80)
PRMstar	20.15 (0.13)	41.66 (36.26)	20.18 (0.18)	41.76 (14.76)	20.09 (0.05)	48.12 (27.74)
LazyPRMstar	20.04 (0.01)	46.70 (7.36)	20.87 (1.52)	37.99 (12.06)	21.91 (6.20)	30.31 (13.98)
FMT	19.34 (1.27)	18.54 (8.12)			15.21 (2.42)	15.31 (8.70)
TRRT	13.14 (5.03)	33.26 (30.84)	7.21 (5.60)	17.36 (2.33)	8.27 (5.15)	13.00 (2.45)
BiTRRT	14.47 (5.82)	29.01 (18.58)			8.46 (8.52)	11.54 (0.48)

Standard deviation in parentheses
Gray cells $\rightarrow$ *time* within $2 \cdot \text{time}_{\text{min}}$ and *path length* within $1.2 \cdot \text{path\_length}_{\text{min}}$, for solved runs $> 80\%$

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 cannot be given. We executed the iterative process to the best of our abilities in order to achieve maximum performance. For planners with an exposed *range* parameter, the distance has to be low enough to provide dense coverage of the configuration space.

Multi-query. For this paper, we only considered single-query motion planning performance. This paper fails to show the potential benefit of lower computing times when using the same roadmap multiple times. For benchmark 1 and 2, we do notice that planners single-query planners are able to compute feasible paths in short amount of time. We therefore argue the need for multi-query planners for online grasp executions similar to benchmarks 1 and 2. The use of multi-query planners can be beneficial when motion constraints have to be used all the time. Computing paths for these motion planning problems can consume more time, as shown in the results. Using the same roadmap again will decrease computing time. However, this map needs to be detailed and the environment needs to be static.

Optimization with time-invariant goal. BiTRRT is the fastest optimizing planner. However, compared to non-optimizing planners, the path length is not consistently shorter. More research needs to be conducted to show the real potential of optimizing planners.

Manipulators. The manipulators studied in this paper have similar specifications. The effect of different manipulator on planner performance cannot be verified with the presented benchmark data. We believe similarly shaped manipulators will yield a similar planner choice. Best overall planner performance, with respect to solved runs, can be obtained with a JACO manipulator.

BFMT and LBTRRT. These planners resulted in errors for the defined motion planning problems. We were unable to provide reliable results to present in this paper. More effort is needed to make these planners work more reliable.

VI. CONCLUSION

This paper presented benchmark data for 21 of the current 23 OMPL planners in MoveIt! for three different manipulators. This data can be useful when performing similar grasp executions. Simultaneously, this paper selected high-performing planners for different motion planning problems, resembling a grasp execution. Planner performance was studied by means of solved runs, computing time and path length. The results showed that the mono-directional KPIECE planner was highest performing when initiating motion planning from a constrained configuration towards a less constrained space. Bi-directional planners with lazy collision-checking (SBL and LBKPIECE) showed fastest performance when the goal configuration is located within a constrained space. Shorter paths were found with LBKPIECE. For motion planning problems incorporating a motion constraints, consistent high performance over the three manipulators was retrieved with BiEST. Considering all the grasp executions presented in this work, RRTConnect was the most reliable planner due to high solved runs. For future work, we would like to investigate the option to implement a faster, easier and more robust method to select parameter values for the planners

REFERENCES

[1] I. Sucan, M. Moll, and L. Kavraki, “Open Motion Planning Library: A Primer,” 2014.

[2] I. A. Sucan and S. Chitta, “MoveIt!” 2013.

[3] Universal Robots, “Technical Specifications UR5,” accessed 2017-03-13. [Online]. Available: https://www.universal-robots.com/

[4] KUKA, “KUKA LWR,” accessed 2017-03-13. [Online]. Available: http://www.kukakore.com/

[5] Kinova Robotics, “Technical Specifications,” accessed 2017-03-13. [Online]. Available: http://www.kinovarobotics.com/

[6] 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.

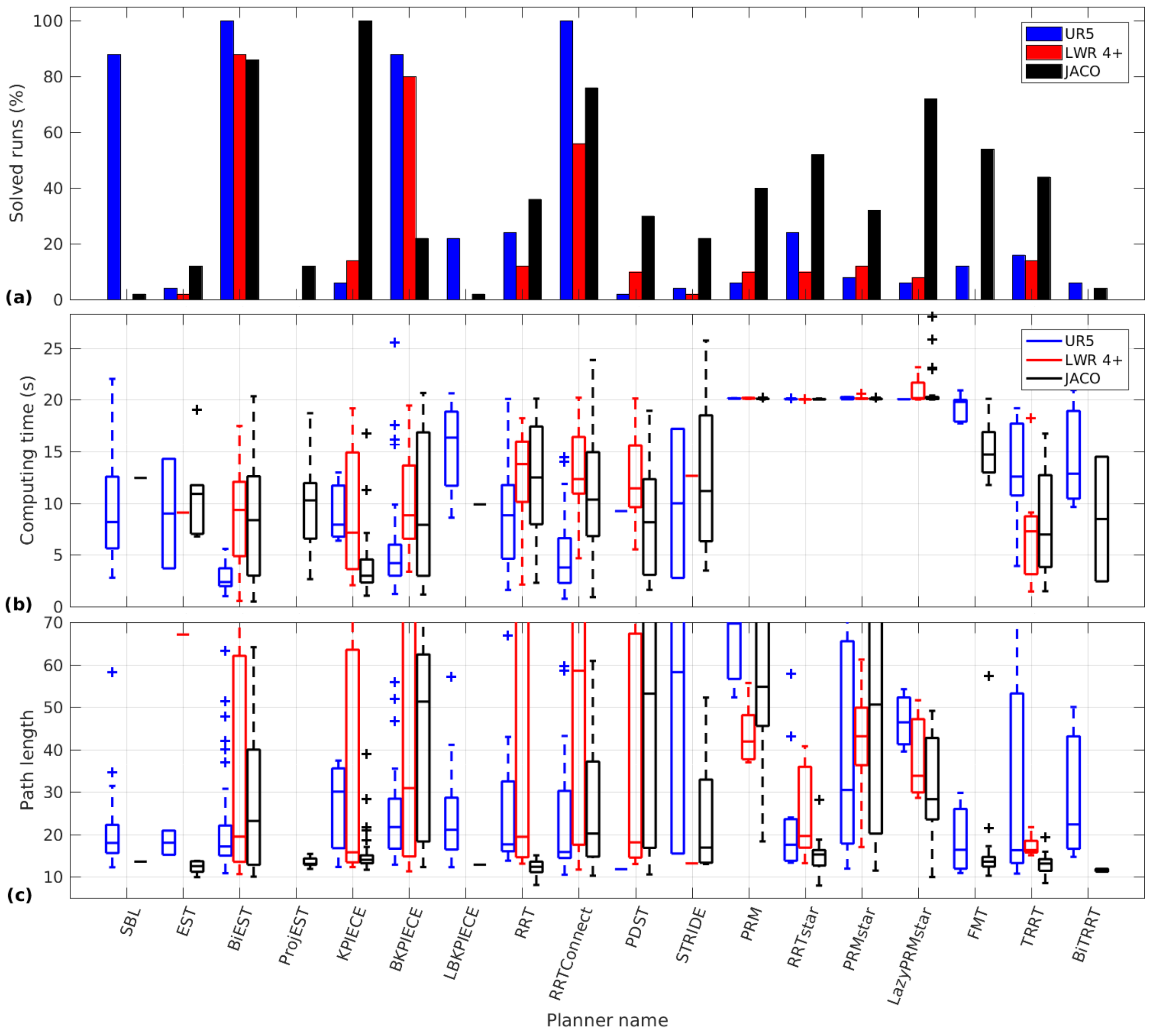

Fig. 6. Results for benchmark 3. (a) Solved runs; higher is better. (b) Computing time; lower is better, small interquartile range is better. (c) Path length; lower is better, small interquartile range is better.

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

[8] I. A. Sucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *In Workshop on the Algorithmic Foundation of Robotics*, 2008.

[9] R. Bohlin and L. E. Kavraki, "Path Planning Using Lazy PRM," in *In IEEE Int. Conf. Robot. & Autom*, 2000, pp. 521–528.

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

[11] J. J. Kuffner 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.

[12] 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.

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

[14] L. Kavraki, P. Svestka, J. claude Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *IEEE Int. Conf. on Robotics and Automation*, 1996, pp. 566–580.

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

[16] 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.

[17] 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.

[18] 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

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

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

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

[22] A. Dobson and K. E. Bekris, "Improving sparse roadmap spanners," Karlsruhe, Germany, 2013.

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