Autonomous Robots (2020) 44:443–467
https://doi.org/10.1007/s10514-019-09832-9

dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning

Rahul Shome[1] · Kiril Solovey[2] · Andrew Dobson[3] · Dan Halperin[2] · Kostas E. Bekris[1]

Received: 1 May 2018 / Accepted: 8 January 2019 / Published online: 24 January 2019

Abstract
Many exciting robotic applications require multiple robots with many degrees of freedom, such as manipulators, to coordinate their motion in a shared workspace. Discovering high-quality paths in such scenarios can be achieved, in principle, by exploring the composite space of all robots. Sampling-based planners do so by building a roadmap or a tree data structure in the corresponding configuration space and can achieve asymptotic optimality. The hardness of motion planning, however, renders the explicit construction of such structures in the composite space of multiple robots impractical. This work proposes a scalable solution for such coupled multi-robot problems, which provides desirable path-quality guarantees and is also computationally efficient. In particular, the proposed dRRT* is an informed, asymptotically-optimal extension of a prior sampling-based multi-robot motion planner, dRRT. The prior approach introduced the idea of building roadmaps for each robot and implicitly searching the tensor product of these structures in the composite space. This work identifies the conditions for convergence to optimal paths in multi-robot problems, which the prior method was not achieving. Building on this analysis, dRRT is first properly adapted so as to achieve the theoretical guarantees and then further extended so as to make use of effective heuristics when searching the composite space of all robots. The case where the various robots share some degrees of freedom is also studied. Evaluation in simulation indicates that the new algorithm, dRRT* converges to high-quality paths quickly and scales to a higher number of robots where various alternatives fail. This work also demonstrates the planner's capability to solve problems involving multiple real-world robotic arms.

Keywords Multi-robot motion planning · Multi-robot problems · Motion planning · Asymptotic optimality · Sampling-based motion planning · Multi-arm motion planning

1 Introduction

A variety of robotic applications, ranging from manufacturing to logistics and service robotics, involve multiple robotic systems operating in the same workspace. In traditional, industrial domains, such as car manufacturing, the environment is fully known and predictable. This allows the robots to operate in a highly scripted manner by repeating the same predefined motions as fast as possible. New types of tasks, however, require robotic manipulators that compute high-quality paths on the fly. For instance, a team of

This is one of the several papers published in *Autonomous Robots* comprising the Special Issue on Multi-Robot and Multi-Agent Systems.

A. Dobson, R. Shome and K. Bekris were supported by NSF IIS 1617744 and CCF 1330789.
K. Solovey and D. Halperin's work has been supported in part by the Israel Science Foundation (Grant No. 825/15) and by the Blavatnik Computer Science Research Fund. Kiril Solovey has also been supported by the Clore Israel Foundation.

✉ Kostas E. Bekris
kostas.bekris@cs.rutgers.com

Rahul Shome
rahul.shome@cs.rutgers.com

Kiril Solovey
kirilsol@post.tau.ac.il

Andrew Dobson
chuples@umich.edu

Dan Halperin
danha@post.tau.ac.il

[1] Computer Science Department of Rutgers University, Piscataway, NJ, USA

[2] Computer Science Department of Tel Aviv University, Tel Aviv, Israel

[3] Electrical Engineering and Computer Science Department of University of Michigan, Ann Arbor, MI, USA

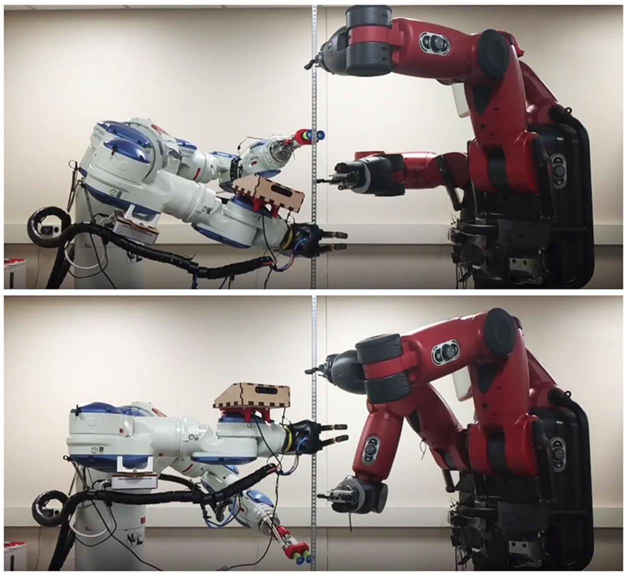


Fig. 1 Planning in a coupled manner for multiple high-dimensional robots is a computationally challenging problem that motivates this work. The image shows an instance of a motion planning problem solved by the proposed approach. It involves 4 robotic arms, each with 7 degrees of freedom, operating in a shared workspace. The arms need to move simultaneously from an initial to a goal configuration

robotic arms can be tasked to pick and sort a variety of objects that are dynamically placed on a common surface. Multiple challenges need to be addressed in the context of such applications, such as detecting the configuration of the objects and grasping. This work deals with the multi-robot motion planning (MMP) problem (Wagner and Choset 2013; Gravot and Alami 2003a; Gharbi et al. 2009) in the context of such setups, i.e., computing the paths of multiple, high-dimensional systems, such as robotic arms, that operate in a shared workspace, as shown in Fig. 1. The focus is to solve MMP in a computationally efficient way as well as in a coupled manner, which allows to argue about the quality of the resulting paths.

Planning for multiple, high-dimensional robotic systems is quite challenging. The motion planning problem is already computationally hard for a single robot (Canny 1988) that is a kinematic chain of rigid bodies. Thus, most approaches for multi-robot motion planning either quickly become intractable as the number of robots increases or alternatively sacrifice completeness and path quality guarantees. In particular, problem instances are especially hard when the robots operate in a shared workspace and in close proximity. In this case, it is not easy to reason for the robots in a decoupled manner. Instead, it is necessary to operate in the composite configuration space of all robots. The space requirements, however, for solving motion planning instances increase exponentially with problem dimensionality. The composite space of all robots in MPP instances is typically very high-dimensional to explore in a comprehensive and resolution complete manner, such as discretizing it with a grid and searching over it.

Sampling-based planners aim to help with such dimensionality issues by approximating the connectivity of the underlying configuration space. They construct graph-based representations, such as a roadmap or a tree data structure, which store collision free configurations and paths through a sampling process. Under certain conditions regarding the density of the corresponding graph, sampling-based planners can provide desirable path quality guarantees. Specifically, they achieve asymptotic optimality, i.e., as the sampling process progresses, the best path on the graph converges to the optimal one in the original configuration space. Nevertheless, even sampling-based planners face significant space challenges in the context of MPP problem, such as the one shown in Fig. 1, which corresponds to a 28-dimensional space. In particular, it becomes infeasible with standard, asymptotically optimal sampling-based planners to explicitly store a graph in the corresponding space that will allow the discovery of a solution in practice. This is due to the large number of samples required to cover an exponentially larger volume as the dimensionality of the underlying space increases. Asymptotically optimal planners must maintain in the order of $logn$ edges per sample, where n is the number of samples. Thus, when planning for high-dimensional systems, the space requirements of the corresponding roadmaps surpass the capabilities of standard workstations rather quickly.

A previously proposed sampling-based planner specifically designed for multi-robot problems, called dRRT (Solovey et al. 2015a), achieved progress in this area by leveraging an implicit representation of the composite space in order to provide both completeness and efficiency. This implicit representation is a graph, which corresponds to the tensor product of roadmaps explicitly constructed for each robot. This allows finding solutions for relatively high-dimensional multi-robot motion planning problems. Nevertheless, this prior method did not provide any path quality guarantees.

One key contribution of this work is to show that the structure of this implicit representation is guaranteed (asymptotically) to contain the optimal path for a set of robots moving simultaneously. Nevertheless, defining an implicit graph that contains a high-quality solution does not guarantee that the final solution is optimal unless the search process over this graph is appropriate. While a provably optimal search approach, such as A^*, could be implemented to search this graph, the extremely large branching factor of the implicit roadmap makes this prohibitively expensive, especially in the context of anytime planning. Instead, this work leverages the observation that a sampling-based method inspired by RRT^*, which maintains a spanning tree over the underlying implicit graph, will return optimal solutions if it allows rewiring operations during the spanning tree construction. Namely, it must converge to the tree with all of the minimum-cost paths starting from the initial query state to each other node in the graph. Further, this work shows that for a broad range of cost functions over paths in this graph can be used while still guaranteeing the proposed `dRRT*` approach will asymptotically converge towards such a tree.

This paper is an extension of prior work (Dobson et al. 2017), which introduced an initial version of the `dRRT*` and the sufficient conditions for generating an asymptotically optimal planner in this context. The current manuscript provides the following extensions:

- A more thorough analysis that shows that the desirable guarantee can be achieved for an additional distance metric for multi-robot motion planning;
- A more detailed description of the method, which has been further improved for computational efficiency purposes through the appropriate incorporation of heuristics;
- The method has been extended to handle systems with shared degrees of freedom, as shown in related work (Shome and Bekris 2017).
- The experimental section has been extended to include the new methods as well as demonstrations on physical platforms.

The following section summarizes related prior work on the subject before Sect. 3 introduces the problem setup. Section 4 describes the underlying structure of the implicit *tensor-roadmap* and the previous method `dRRT` (Solovey et al. 2015a). The changes to `dRRT` necessary to achieve asymptotic optimality and computational efficiency, which result to the proposed algorithm `dRRT*` are presented in Sect. 5. An analysis of the properties of the method are showcased in Sect. 6. The method is extended, in Sect. 7 to systems with shared degrees of freedom. Section 8 evaluates the methods experimentally and demonstrates their performance.

2 Prior work

The multi-robot motion planning problem (MMP) is notoriously difficult as it involves many degrees of freedom, and consequently a vast search space, as each additional robot introduces several additional degrees of freedom to the problem. Certain instances of the problem can be solved efficiently, i.e., in polynomial run time, and in a complete manner, at times even with optimality guarantees on the solution costs (Turpin et al. 2013; Adler et al. 2015; Solovey et al. 2015b). However, in general MMP is computationally intractable (Hopcroft et al. 1984; Spirakis and Yap 1984; Solovey and Halperin 2016; Johnson 2018).

Decoupled MMP techniques (Erdmann and Lozano-Perez 1987; Ghrist et al. 2005; LaValle and Hutchinson 1998; Peng and Akella 2004; Van Den Berg and Overmars 2005; Van Den Berg et al. 2009) reduce search space size by partitioning the problem into several subproblems, which are solved separately. Then, the different solutions are combined. These methods, however, typically lack completeness and optimality guarantees. While some hybrid approaches can take advantage of the inherent decoupling between robots and provide guarantees (Van Den Berg et al. 2009), they are often limited to discrete domains. The problem is more complex when the robots exhibit non-trivial dynamics (Peng and Akella 2005). Collision avoidance or control methods can scale to many robots, but lack path quality guarantees (Van Den Berg et al. 2011; Tang and Kumar 2015).

In contrast to that, centralized approaches (Kloder and Hutchinson 2005; O'Donnell and Lozano-Pérez 1989; Salzman et al. 2015; Solovey et al. 2015a; Svestka and Overmars 1998; Wagner and Choset 2013) usually work in the combined high-dimensional configuration space, and thus tend to be slower than decoupled techniques. However, centralized algorithms often come with stronger theoretical guarantees, such as completeness. Through the rest of this section we will consider centralized methods, with an emphasis on sampling-based approaches.

Sampling-based algorithms for a single robot (Kavraki et al. 1996; LaValle and Kuffner 1999; Karaman and Frazzoli 2011) can be extended to the multi-robot case by considering the fleet of robots as one composite robot (Gildardo 2002). Such an approach suffers from inefficiency as it overlooks aspects of multi-robot planning, and hence can handle only a very small number of robots. Several techniques tailored for instances of MMP involving a small number of robots have been described (Hirsch and Halperin 2004; Salzman et al. 2015).

In previous work (Solovey and Halperin 2014), an extension of MMP was introduced, which consists of several groups of interchangeable robots. At the heart of the algorithm is a novel technique where the problem is reduced to several discrete pebble-motion problems (Kornhauser et al. 1984; Luna and Bekris 2011; Yu and LaValle 2013). These reductions amplify basic samples into massive collections of free placements and paths for the robots. An improved version (Krontiris et al. 2015) of this algorithm applied it to rearrange multiple objects using a robotic manipulator.

Previous work (Svestka and Overmars 1998) introduced a different approach, which leverages the following fundamental observation: the structure of the overall high-dimensional multi-robot configuration space can be inferred by first considering independently the free space of every robot, and combining these subspaces in a meaningful manner to account for robot-robot collisions. They suggested an approach which combines roadmaps constructed for individual robots into one *tensor-product* roadmap $\hat{\mathbb{G}}$, which captures the structure of the joint configuration space (see more information in Sect. 4).

Due to the exponential nature of the resulting roadmap, this technique is only applicable to problems that involve a modest number of robots. A recent work (Wagner and Choset 2013) suggests that $\hat{\mathbb{G}}$ does not necessarily have to be explicitly represented. They apply their M* algorithm to efficiently retrieve paths over $\hat{\mathbb{G}}$, while minimizing the explored portion of the roadmap. The resulting technique is able to cope with a large number of robots, for certain types of scenarios. However, when the degree of simultaneous coordination between the robots increases, there is a sharp increase in the running time of this algorithm, as it has to consider many neighbors of a visited vertex of $\hat{\mathbb{G}}$. This makes M*less effective when the motion of multiple robots needs to be tightly coordinated.

Recently a different sampling-based framework for MMP was introduced, which combines an implicit representation of $\hat{\mathbb{G}}$ with a novel approach for pathfinding in geometrically-embedded graphs tailored for MMP (Solovey et al. 2015a). The *discrete-RRT* (dRRT) algorithm is an adaptation of the celebrated RRT algorithm for the discrete case of a graph, and it enables a rapid exploration of the high-dimensional configuration space by carefully walking through an implicit representation of the tensor product of roadmaps for the individual robots (see extensive description in Sect. 4). The approach was demonstrated experimentally on scenarios that involve as many as 60 DoFs and on scenarios that require tight coordination between robots. On most of these scenarios dRRT was faster by a factor of at least ten when compared to existing algorithms,including the aforementioned M*.

Later, dRRT was applied to motion planning of a free-flying multi-link robot (Salzman et al. 2016). In that case, dRRT allowed to efficiently decouple between costly self-collision checks, which were done offline, and robot-obstacle collision checks, by traversing an implicitly-defined roadmap, whose structure resembles to that of $\hat{\mathbb{G}}$. dRRT has also been used in the study of the effectiveness of metrics for MMP, which are an essential ingredient in sampling-based planners (Atias et al. 2017).

The current work proposes dRRT* and shows that it is an efficient asymptotically optimal extension of the previously proposed dRRT. The dRRT* framework is an anytime algorithm, which quickly finds initial solutions and then refines them, while ensuring asymptotic convergence to optimal solutions. Simulations show that the method practically generates high-quality paths while scaling to complex, high-dimensional problems, where alternatives fail.

3 Problem setup and notation

We start with a definition of the problem. Consider a shared workspace with $R \geq 2$ holonomic robots, each operating in a d-dimensional configuration space $\mathbb{C}_i \subset \mathbb{R}^d$ for $1 \leq i \leq R$. For a given robot i, denote its free space, i.e., the set of all collision free configurations, by $\mathbb{C}_i^{\mathrm{f}} \subset \mathbb{C}_i$, and the obstacle space by $\mathbb{C}_i^{\mathrm{o}} = \mathbb{C}_i \setminus \mathbb{C}_i^{\mathrm{f}}$.

The *composite configuration space* $\mathbb{C} = \prod_{i=1}^{R} \mathbb{C}_i$ is the Cartesian product of each robot's configuration space. That is, a composite configuration $Q = (q_1, \ldots, q_R) \in \mathbb{C}$ is an R-tuple of robot configurations. For two distinct robots i, j, denote by $I_i^j(q_j) \subset \mathbb{C}_i$ the set of configurations of robot i, which lead into collision with robot j at its configuration q_j. Then, the composite free space $\mathbb{C}^{\mathrm{f}} \subset \mathbb{C}$ consists of configurations $Q = (q_1, \ldots, q_R)$ in which robots do not collide with obstacles or pairwise with each other. Formally:

- $q_i \in \mathbb{C}_i^{\mathrm{f}}$ for every $1 \leq i \leq R$;
- $q_i \notin I_i^j(q_j), q_j \notin I_j^i(q_i)$ for every $1 \leq i < j \leq R$.

The composite obstacle space is defined as $\mathbb{C}^{\mathrm{o}} = \mathbb{C} \setminus \mathbb{C}^{\mathrm{f}}$.

Multi-robot planning is concerned with finding (collision-free) composite trajectories of the form $\Sigma : [0, 1] \to \mathbb{C}^{\mathrm{f}}$. Σ is an R-tuple $(\sigma_1, \ldots, \sigma_R)$ of single-robot trajectories $\sigma_i : [0, 1] \to \mathbb{C}_i$.

This work is concerned with producing high-quality trajectories, which minimize certain *cost functions*. In particular, we consider three cost functions cost$(\cdot)$, which are presented below. Let $\Sigma = (\sigma_1, \ldots, \sigma_R)$ be a composite trajectory. For the following, $\|\cdot\|$ denotes the standard *arc length* of a curve:

- The sum of path lengths: $\mathrm{cost}(\Sigma) = \sum_{i=1}^{R} \|\sigma_i\|$.
- The maximum path length: $\mathrm{cost}(\Sigma) = \max_{i=1:R} \|\sigma_i\|$.
- The Euclidean arc length of $\Sigma : \mathrm{cost}(\Sigma) = \|\Sigma\|$

This work presents dRRT* as an efficient, anytime solution to the robustly-feasible composite motion planning (RFCMP) problem:

Definition 1 (*RFCMP*) Given R robots operating in composite configuration space $\mathbb{C} = \prod_{i=1}^{R} \mathbb{C}_i$, and for a given query $S = (s_1, \ldots, s_R), T = (t_1, \ldots, t_R)$, an RFCMP problem is one which yields a robustly-feasible trajectory $\Sigma : [0, 1] \to \mathbb{C}^{\mathrm{f}}$ and $\Sigma(0) = S, \Sigma(1) = T$. Namely, there exists a fixed constant $\delta > 0$ such that $\forall\ \tau \in [0, 1], X \in \mathbb{C}^{\mathrm{o}}$ it holds that

$$\|\Sigma(\tau) - X\| \geq \delta.$$

One of the primary objectives of this work is to provide asymptotic optimality in the composite configuration space

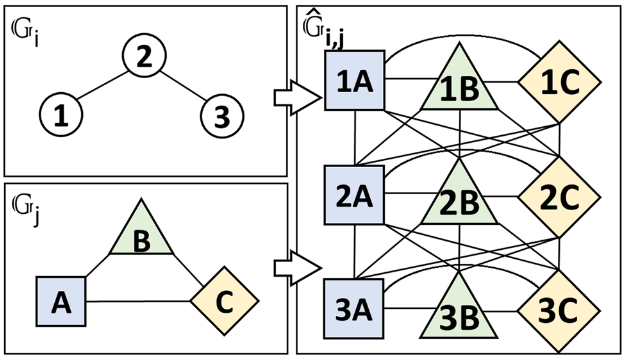


Fig. 2 Given the individual robot roadmaps $\mathbb{G}_i$ and $\mathbb{G}_j$ shown on the left, the tensor product roadmap $\hat{\mathbb{G}}_{i,j}$ arises, which is shown on the right. For each pair of nodes, where one is selected from $\mathbb{G}_i$ and the other from $\mathbb{G}_j$, a node is defined in $\hat{\mathbb{G}}_{i,j}$. Two nodes in the tensor-product roadmap share an edge if their constituent nodes in the individual robot roadmaps also share an edge. For instance, nodes $1A$ and $3C$ do not share an edge in $\hat{\mathbb{G}}_{i,j}$ because nodes 1 and 3 do not share an edge in $\mathbb{G}_i$

without explicitly constructing a planning structure in this space.

Definition 2 (*Asymptotic Optimality*) Let m be the time budget of the algorithm and a *robustly optimal* solution $\Sigma^{(m)}$ of cost c^* is returned after time m, then asymptotic optimality is defined as ensuring that the following holds true for any $\epsilon > 0$.

$$\lim_{m\to\infty} \Pr\left[\text{cost}(\Sigma^{(m)}) \le (1+\epsilon)c^*\right] = 1.$$

4 Algorithmic foundations

This section provides a detailed description of the *discrete*-RRT (dRRT) method (Solovey et al. 2015a), which is the basis of our method presented in Sect. 5. dRRT was posed as an efficient way to search an implicitly defined tensor-product roadmap, which captures the structure of $\mathbb{C}$ without explicitly sampling this space.

4.1 Tensor-product roadmap

Here we provide a formal definition of the tensor-product roadmap that dRRT is designed to explore. For every robot $1 \le i \le R$ construct a PRM graph (Kavraki et al. 1996), denoted by $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$, which is embedded in $\mathbb{C}_i^{\text{f}}$. That is, $\mathbb{G}_i$ can be viewed as an approximation of $\mathbb{C}_i^{\text{f}}$ and encodes collision free motions for robot i. The construction of $\mathbb{G}_i$ is determined by two parameters n and r_n, which represent the number of samples, and the connection radius, respectively. As will be discussed in the following sections, it is necessary the roadmaps $\mathbb{G}_1, \ldots, \mathbb{G}_R$ to be constructed with certain range of parameters to guarantee asymptotic optimality of the new planners (Sect. 5).

Define the *tensor-product roadmap*, denoted by $\hat{\mathbb{G}} = (\hat{\mathbb{V}}, \hat{\mathbb{E}})$, as the tensor product between $\mathbb{G}_1, \ldots, \mathbb{G}_R$ (see Fig. 2). Each vertex of $\hat{\mathbb{G}}$ describes a simultaneous placement of the R robots, and similarly an edge of $\hat{\mathbb{G}}$ describes a simultaneous motion of the robots. Formally, $\hat{\mathbb{V}} = \{(v_1, v_2, \ldots, v_R) : \forall\ i,\ v_i \in \mathbb{V}_i\}$ is the Cartesian product of the nodes from each roadmap $\mathbb{G}_i$. For two vertices $V = (v_1, \ldots, v_m) \in \hat{\mathbb{V}}$, $V' = (v'_1, \ldots, v'_m) \in \hat{\mathbb{V}}$, the edge set $\hat{\mathbb{E}}$ contains edge (V, V') if $\forall i \in [1, R] :\ v_i = v'_i$ or $(v_i, v'_i) \in \mathbb{E}_i$.[1] Note that by the definition of $\mathbb{G}_1, \ldots, \mathbb{G}_R$, the motion described by each edge $E \in \hat{\mathbb{E}}$ represents a path for the R robots in which the robots do not collide with obstacles. However, collisions between pairs of robots still may be possible.

It is important to note that the *tensor-product roadmap* has $\|\hat{\mathbb{V}}\| = \prod_{i=1}^{R} \|\hat{\mathbb{V}}_i\|$ vertices. Given the neighborhood of a node v_i in $\mathbb{G}_i$ as $\texttt{Adj}(v_i, \mathbb{G}_i)$, the size of the neighborhood of a node $v = \{v_1 \ldots v_R\}$ in $\hat{\mathbb{G}}$ is $\|\texttt{Adj}(v, \hat{\mathbb{G}})\| = \prod_{i=1}^{R} \|\texttt{Adj}(v_i, \mathbb{G}_i)\|$. Using the much smaller $\mathbb{G}_1, \ldots, \mathbb{G}_R$ to construct $\hat{\mathbb{G}}$ online is computationally beneficial.

The presented algorithms share a common set of input and output parameters, such as the configuration space decompositions, which are predefined. In practice, the algorithms use pre-computed roadmaps in each constituent space online. The collision volumes that correspond to the robot and obstacles in the scene are also used online for validation. The algorithms output a trajectory in the configuration space of all robots, which is collision free with all obstacles and among robots.

4.2 Discrete RRT

An explicit construction of $\hat{\mathbb{G}}$ is possible in very limited settings that either involve few robots, e.g., $R = 2$, or when the underlying single-robot roadmaps have few vertices and edges. However, in general it is prohibitively costly to fully represent it due to its size, which grows exponentially with the number of robots, in terms of the number of vertices. Moreover, in some cases it may be even a challenge to represent all the edges adjacent to a single vertex of $\hat{\mathbb{G}}$, as there may be exponentially many of those.

The dRRT algorithm enjoys the rich structure that $\hat{\mathbb{G}}$ offers (see Sect. 6) without explicitly representing it. In particular, it gathers information on $\hat{\mathbb{G}}$ only from the single-robot roadmaps $\mathbb{G}_1, \ldots, \mathbb{G}_R$.

Similarly to the single-robot planner RRT (LaValle and Kuffner 1999), dRRT grows a tree rooted at the start state

[1] Notice this difference from the original dRRT (Solovey et al. 2015a) so as to allow edges where some robots remain motionless.

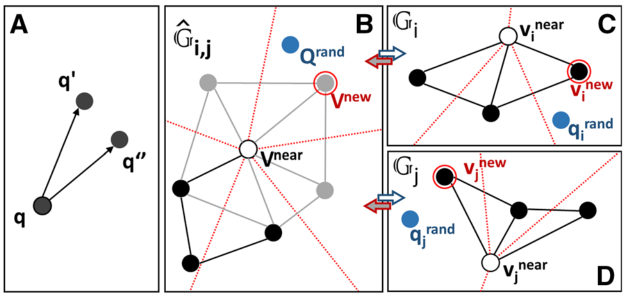


Fig. 3 **a** The method reasons over all neighbors q' of q so as to minimize the angle $\measuredangle_q(q', q'')$. **b** $\mathbb{O}_d(\cdot,\cdot)$ finds graph vertex V^{new} by minimizing angle $\measuredangle_{V^{near}}(V^{new}, Q^{rand})$. **c, d** $V^{near}$and $Q^{rand}$are projected into each robot's $\mathbb{C}$-space to find v_i^{new} and v_j^{new}, respectively, which minimize both $\measuredangle_{v_i^{near}}(v_i^{new}, q_i^{rand})$ and $\measuredangle_{v_j^{near}}(v_j^{new}, q_j^{rand})$

of the given query (Line 1). dRRT restricts the growth of its tree $\mathbb{T}$ to the tensor-product roadmap $\hat{\mathbb{G}}$ in contrast to RRT, which explores the entire space $\mathbb{C}$. That is, $\mathbb{T}$ is a subgraph of $\hat{\mathbb{G}}$, and $\mathbb{T} \subset \mathbb{C}^f$.

The high level operations of the dRRT approach are outlined in Algorithm 1. The approach will iterate until a solution is found or the time limit is exceeded (Algorithm 1, Line 2), beginning with performing a fixed number n_{it} expansion steps at the beginning of each iteration (Lines 3, 4). This expansion process is outlined in Algorithm 2. The approach then checks to see if there is a connected path to the target (Line 5), and once a path is found, it is returned (Lines 6, 7).

The expansion procedure begins by drawing a random sample $Q^{rand} \in \mathbb{C}$ (Line 1). It then finds the nearest neighbor V^{near} in the tree (Line 2) and then selects a neighbor V^{new}, such that $(V^{near}, V^{new}) \in \hat{\mathbb{E}}$, according to a *direction oracle* function $\mathbb{O}_d$ (Line 3). Then, if V^{new} is not in the tree (Line 4), it is added to the tree (Line 5) and an edge from V^{near} to V^{new} is also added (Line 6).

We now elaborate on $\mathbb{O}_d$. Given V^{near}, Q^{rand}, the oracle returns a vertex $V^{new} \in \hat{\mathbb{V}}$ that is the neighbor of V^{near} (in $\hat{\mathbb{G}}$) found in the direction of Q^{rand}. The crux of the approach is that $\mathbb{O}_d$ can come up with such a neighbor efficiently without relying on explicit representation of $\hat{\mathbb{G}}$. Let $Q, Q', Q'' \in \mathbb{C}$ and define $\rho(Q, Q')$ to be the ray through Q' starting at Q. Then, denote $\measuredangle_Q(Q', Q'')$ as the minimum angle between $\rho(Q, Q')$ and $\rho(Q, Q'')$. Denote by $\texttt{Adj}(V^{near}, \hat{\mathbb{G}})$ the set of neighbor nodes of V^{near} in $\hat{\mathbb{G}}$, i.e., for every $V \in \texttt{Adj}(V^{near}, \hat{\mathbb{G}})$ it holds that $(V^{near}, V) \in \hat{\mathbb{E}}$. Then

$$\mathbb{O}_d(V^{near}, Q^{rand}) = \operatorname*{argmin}_{V \in \texttt{Adj}(V^{near}, \hat{\mathbb{G}})} \measuredangle_{V^{near}}(Q^{rand}, V).$$

The implementation of $\mathbb{O}_d$ (Algorithm 5) proceeds in the following manner (see a two-robot case illustrated in Fig. 3). Let $Q^{rand} = (q_1^{rand}, \ldots, q_R^{rand})$, $V^{near} = (v_1^{near}, \ldots, v_R^{near})$.

Algorithm 1: dRRT($\hat{\mathbb{G}}, S, T, n_{it}$)
```
1 $\mathbb{T}$.init($S$);
2 while time.elapsed() < time_limit do
3     for $i : 1 \to n_{it}$ do
4         Expand($\hat{\mathbb{G}}, \mathbb{T}$);
5     $\pi \leftarrow$ Connect_to_Target($\hat{\mathbb{G}}, \mathbb{T}, T$);
6     if $\pi \neq \emptyset$ then
7         return Trace_Path($\mathbb{T}, T$);
8 return $\emptyset$
```

Algorithm 2: Expand($\hat{\mathbb{G}}, \mathbb{T}$)
```
1 $Q^{rand} \leftarrow$ Random_Sample();
2 $V^{near} \leftarrow$ Nearest_Neighbor($\mathbb{T}, Q^{rand}$);
3 $V^{new} \leftarrow \mathbb{O}_d(V^{near}, Q^{rand})$;
4 if $V^{new} \notin \mathbb{T}$ then
5     $\mathbb{T}$.Add_Vertex($V^{new}$);
6     $\mathbb{T}$.Add_Edge($V^{near}, V^{new}$);
```

For every robot $1 \leq i \leq R$, the oracle extracts from $\mathbb{G}_i$ the neighbor v_i^{new} of v_i^{near}, which minimizes the expression $\measuredangle_{v_i^{near}}(q_i^{rand}, v_i^{new})$. Notice that such a search can be performed efficiently as it only requires to traverse all the neighbors of v_i^{near} in $\mathbb{G}_i$. The combination of all v_i^{near} yields V^{near}.

As in RRT, dRRT has a Voronoi-bias property (Lindemann and LaValle 2004). Showing that dRRT exhibits Voronoi bias is slightly more involved compared to the basic RRT. This is illustrated in Fig. 4. To generate an edge (V, V^{new}), random sample Q^{rand} must be drawn within the Voronoi cell of V, denoted as Vor(V) (Fig. 4a) and in the general direction of V^{new}, denoted as Vor$'(V)$ (Fig. 4b). The intersection of these two volumes: $\text{Vol}(V) = \text{Vor}(V) \cap \text{Vor}'(V)$, is the volume to be sampled so as to generate V^{new}via V^{near} as shown in Fig. 4.

The high-level loop of the algorithm remains similar across the method variants. The input parameter n_{it} denotes how many times the tree is expanded before the algorithm checks whether a solution has already been discovered. If $n_{it} = 1$, this check is performed every iteration. If tracing the path is an expensive operation—typically it corresponds to a heuristic search process over the tensor product roadmap—then the implementer can choose to use a higher value.

5 Asymptotically optimal discrete RRT

This section outlines two versions of the proposed asymptotically optimal variant of the dRRT method. The first is a simple uninformed approach, which relies on the fact that to provide asymptotic optimality, it is sufficient to use a simple rewiring scheme. This simplified version will be called the asymptotically-optimal dRRT (ao-dRRT). For the sake

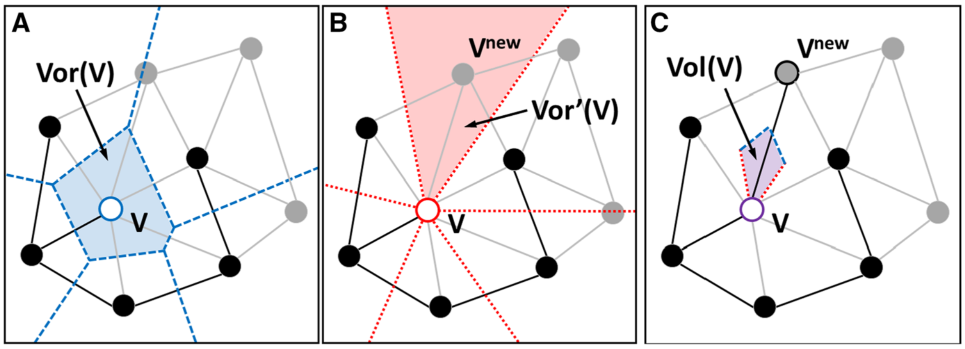


Fig. 4 **a** The Voronoi region Vor(V) of vertex V is shown where if Q^{rand} is drawn, vertex V is selected for expansion. **b** When Q^{rand} lies in the directional Voronoi region Vor$'(V)$, the expand step expands to V^{new}. **c** Thus, when Q^{rand} is drawn within volume Vol$(V) = $ Vor$(V) \cap$ Vor$'(V)$, the method will generate V^{new} via V

of algorithmic efficiency however, a second, more advanced version is also proposed referred to as dRRT*. To summarize the algorithmic contributions of the current work over the original dRRT:

- dRRT* performs a rewiring step to refine paths in the tree, reducing costs to reach particular nodes.
- dRRT* is anytime, employing branch and bound pruning after an initial solution is reached.
- dRRT* promotes progress towards the goal during tree node selection.
- dRRT* employs an informed expansion procedure $\mathbb{I}_d$ capable of using heuristic guidance.

5.1 ao-dRRT

This section outlines ao-dRRT, an asymptotically optimal version of the dRRT algorithm which has been minimally modified to guarantee asymptotic optimality. At a high-level the approach uses a tree re-wiring technique reminiscent of RRT* (Karaman and Frazzoli 2011).

Algorithm 3 outlines ao-dRRT which iteratively expands a tree $\mathbb{T}$ over $\hat{\mathbb{G}}$ given a time budget (Algorithm 3, Line 2), performing n_{it} consecutive calls to Expand_ao-dRRT (Lines 3, 4). Then, the method attempts to trace the path π which connects the start S with the target T (Line 5). If such a path is found and is better than π_{best}, it replaces π_{best} (Lines 6, 7). π_{best} is returned after the time limit is reached (Line 8).

Algorithm 3: ao-dRRT($\hat{\mathbb{G}}, S, T, n_{\text{it}}$)

```
1 $\pi_{\text{best}} \leftarrow \emptyset$, $\mathbb{T}$.init($S$);
2 while time.elapsed() < time_limit do
3     for $i : 1 \rightarrow n_{it}$ do
4         Expand_ao-dRRT($\hat{\mathbb{G}}, \mathbb{T}$);
5     $\pi \leftarrow$ Connect_to_Target($\hat{\mathbb{G}}, \mathbb{T}, T$);
6     if $\pi \neq \emptyset \cap \text{cost}(\pi) < \text{cost}(\pi_{\text{best}})$ then
7         $\pi_{\text{best}} \leftarrow$ Trace_Path($\mathbb{T}, T$)
8 return $\pi_{\text{best}}$
```

Algorithm 4: Expand_ao-dRRT($\hat{\mathbb{G}}, \mathbb{T}$)

```
1 $Q^{\text{rand}} \leftarrow$ Random_Sample();
2 $V^{\text{near}} \leftarrow$ Nearest_Neighbor($\mathbb{T}, Q^{\text{rand}}$);
3 $V^{\text{new}} \leftarrow \mathbb{O}_d(V^{\text{near}}, Q^{\text{rand}})$;
4 if $V^{\text{new}} \notin \mathbb{T}$ then
5     $\mathbb{T}$.Add_Vertex($V^{\text{new}}$);
6     $\mathbb{T}$.Add_Edge($V^{\text{near}}, V^{\text{new}}$);
7 else
8     $\mathbb{T}$.Rewire($V^{\text{near}}, V^{\text{new}}$);
```

Algorithm 5: $\mathbb{O}_d(V^{\text{near}}, Q^{\text{rand}}, \hat{\mathbb{G}})$

```
1 for $i : 1 \rightarrow R$ do
2     $v_i^{\text{new}} \leftarrow \underset{v \in \text{Adj}(v_i^{\text{near}}, \mathbb{G}_i)}{argmin} \angle_{v_i^{\text{near}}}(q_i^{\text{rand}}, v)$ ;
3 return $V^{\text{new}}$
```

The expansion procedure for ao-dRRT is very similar to the original dRRT method, and is outlined in Algorithm 4. It begins by drawing a random sample in the composite configuration space (Line 1), and then finds the nearest neighbor V^{near} to this sample in the tree (Line 2). It then selects a neighbor V^{new} according to the oracle function $\mathbb{O}_d$ (Algorithm 5). This is the same oracle that is used in dRRT that tries to select a neighbor of V^{near} most in the direction of Q^{rand}. Then, if V^{new} is not in the tree (Line 4), it is added to the tree (Line 5) and an edge from V^{near} to V^{new} is also added (Line 6). Where this expansion step differs is that if V^{new} is already in the tree (Line 7), the method performs a rewiring step (Line 8) to check to see if the path to V^{new} is of lower cost than the existing one.

The method would be similar to dRRT in terms of the samples that constitute the tree, however ao-dRRT improves the solution cost with iterations and finds better solutions compared to dRRT. It is however desirable to focus the search in order find the initial solution quickly, while preserving solution quality improvement over time.

5.2 dRRT*

The main body of the informed dRRT* algorithm is provided in Algorithm 6. The proposed method is an improvement on top of ao-dRRT, that preserves the asymptotic optimality while benefiting computationally from *branch-and-bound* pruning once a solution is found, greedy child propagations during node selection, and heuristic guidance during expansions.

The key insight behind the algorithmic improvements is the fact that by virtue of the structure of the *tensor-product roadmap* $\hat{\mathbb{G}}$, there readily exists a usable heuristic measure $\mathbb{H}$ in the constituent roadmaps $\mathbb{G}_i$. The shortest path on a constituent roadmap to the goal T can be used as a heuristic to guide the tree.

If there is no robot interaction introduced by the individual robot shortest paths, such a path comprising of the individual shortest paths is a solution that suffices. In cases of interaction between the robots, a shortest path is expected to deviate locally in regions of interaction. The best a robot can do from any constituent roadmap vertex is to follow the shortest path to the goal on the constituent roadmap. Although domain-specific heuristics can be also applied to the algorithm, it should be noted that in the currently proposed method, the purpose of the heuristic is to primarily discover an initial solution as quickly as possible. This in turn helps *branch-and-bound* kick in and further focuses the search once a bounding cost is ascertained from the initial solution.

At a high level, Algorithm 6 follows the structure of ao-dRRT. The only change is that the outer loop keeps track of the tree node being added as V^{last} and passes it on to the next call to the Expand_dRRT* subroutine. The use of this information to apply heuristic guidance is detailed in the description of the function.

Algorithm 7 outlines the expansion step. The default behavior is summarized in Algorithm 7, Lines 1–3, i.e., when no V^{last} is passed as argument (Line 1). This operation corresponds to an exploration step similar to RRT, i.e., a random sample Q^{rand} is generated in $\mathbb{C}$ (Line 2) and its nearest neighbor V^{near} in $\mathbb{T}$ is found (Line 3). If the last iteration generated a node V^{last} that was closer to the goal relative to its parent, then V^{last} is provided to the function. In this case (Line 4–6) Q^{rand} is set as the target configuration T, and V^{near} is selected as V^{last}. This constitutes the greedy child propagations which promotes progress towards the goal.

Algorithm 6: dRRT*($\hat{\mathbb{G}}$, S, T, n_{it})
```
1 $\pi_{\text{best}} \leftarrow \emptyset$, $\mathbb{T}$.init($S$), $V^{\text{last}} \leftarrow S$;
2 while time.elapsed() < time_limit do
3     for $i : 1 \rightarrow n_{it}$ do
4         $V^{\text{last}} \leftarrow$ Expand_dRRT*($\hat{\mathbb{G}}$, $\mathbb{T}$, $V^{\text{last}}$, $T$);
5     $\pi \leftarrow$ Connect_to_Target($\hat{\mathbb{G}}$, $\mathbb{T}$, $T$);
6     if $\pi \neq \emptyset \cap \text{cost}(\pi) < \text{cost}(\pi_{\text{best}})$ then
7         $\pi_{\text{best}} \leftarrow$ Trace_Path($\mathbb{T}$, $T$)
8 return $\pi_{\text{best}}$
```

Algorithm 7: Expand_dRRT*($\hat{\mathbb{G}}$, $\mathbb{T}$, V^{last}, T)
```
1 if $V^{\text{last}} = \emptyset$ then
2     $Q^{\text{rand}} \leftarrow$ Random_Sample();
3     $V^{\text{near}} \leftarrow$ Nearest_Neighbor($\mathbb{T}$, $Q^{\text{rand}}$);
4 else
5     $Q^{\text{rand}} \leftarrow T$;
6     $V^{\text{near}} \leftarrow V^{\text{last}}$;
7 $V^{\text{new}} \leftarrow \mathbb{I}_{\text{d}}(V^{\text{near}}, Q^{\text{rand}}, \hat{\mathbb{G}}, T)$;
8 $N \leftarrow$ Adj($V^{\text{new}}$, $\hat{\mathbb{G}}$) $\cap\ \mathbb{T}$;
9 $V^{\text{best}} \leftarrow \underset{V \in N}{argmin}\ \text{cost}(V) + \text{cost}(\mathbb{L}(V, V^{\text{new}}))$;
10 if $V^{\text{best}} = \emptyset$ then return $\emptyset$;
11 if $\text{cost}(V^{\text{new}}) > \text{cost}(\pi_{\text{best}})$ then return $\emptyset$;
12 if $V^{\text{new}} \notin \mathbb{T}$ then
13     $\mathbb{T}$.Add_Vertex($V^{\text{new}}$);
14     $\mathbb{T}$.Add_Edge($V^{\text{best}}$, $V^{\text{new}}$);
15 else $\mathbb{T}$.Rewire($V^{\text{best}}$, $V^{\text{new}}$) ;
16 for $V \in N$ do
17     if $\text{cost}(V^{\text{new}}) + \text{cost}(\mathbb{L}(V^{\text{new}}, V)) <$
       $\text{cost}(V) \cap \mathbb{L}(V^{\text{new}}, V) \subset \mathbb{C}^{\text{f}}$ then
18         $\mathbb{T}$.Rewire($V^{\text{new}}$, $V$);
19 if $\mathbb{H}(V^{\text{new}}) < \mathbb{H}(V^{\text{best}})$ then
20     return $V^{\text{new}}$;
21 else return $\emptyset$
```

Algorithm 8: $\mathbb{I}_{\text{d}}(V^{\text{near}}, Q^{\text{rand}}, \hat{\mathbb{G}}, T)$
```
1 for $i : 1 \rightarrow R$ do
2     if $q_i^{\text{rand}} = T_i$ then
3         $v_i^{\text{new}} \leftarrow \underset{x \in \text{Adj}(v_i^{\text{near}}, \mathbb{G}_i)}{argmin}\ \mathbb{H}(x, T_i, \mathbb{G}_i)$ ;
4     else
5         $v_i^{\text{new}} \leftarrow$ random(Adj($v_i^{\text{near}}$, $\mathbb{G}_i$));
6 return $V^{\text{new}}$
```

Informed expansion $\mathbb{I}_d$ The expansion procedure in Algorithm 8 replaces the oracle in dRRT. It switches between distinct guided and exploratory behaviors according to whether q_i^{rand} attempts to drive the expansion towards the target T or not. When the method uses heuristic guidance (Fig. 5), among all the neighbors of v_i^{near} on a constituent roadmap $\mathbb{G}_i$, Adj(v_i^{near}, $\mathbb{G}_i$), the one with the best heuristic measure $\mathbb{H}$ is selected. During the exploration phase, the method selects a random neighbor out of Adj(v_i^{near}, $\mathbb{G}_i$).

In either case, the oracle function $\mathbb{I}_d$ returns to Algorithm 7 the implicit graph node V^{new} that is a neighbor of V^{near} on the implicit graph (Line 7). Then the method finds neighbors N, which are adjacent to V^{new} in $\hat{\mathbb{G}}$ and have also been added to $\mathbb{T}$ (Line 8). Among N, the best node V^{best} is chosen as the node to connect V^{new} according to cost measure. Such an operation might yield no valid parent V^{best} due to collisions along $\mathbb{L}(\cdot)$. In such a case (Line 10) the method fails to add

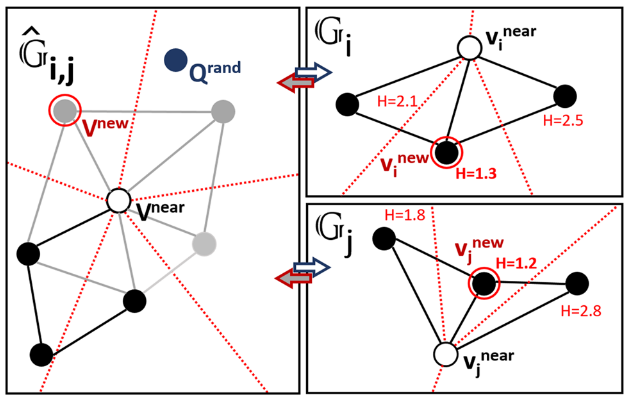


Fig. 5 The method selects a neighbor with the best heuristic estimate $\mathbb{H}$ from each of the constituent roadmaps during the guided expansion phase in $\mathbb{I}_d$

a node during the current iteration. Line 11 implements a *branch-and-bound* based on the cost of the best solution so far.

Lines 12–15 recount the tree addition and parent rewire process. Lines 16–17 perform an additional rewiring step in the neighborhood if V^{new} is a better parent of any of the neighboring nodes. Line 19 switches the child promotion by checking whether V^{new} made progress toward the goal according to the heuristic measure. The method ensures that in this case V^{new}(or V^{last}) is a child-promoted node, which would be selected during the next iteration. This effect of this behavior is that if the uncoordinated individual shortest paths are collision free, this would be greedily attempted first from the child-promoted nodes added to the tree. Evaluations indicate that this proves very effective in practice.

It should be noted that all candidate edges $\mathbb{L}(\cdot)$ in Line 9 and 17 are collision-free and for the sake of algorithmic clarity, collision checking has been assumed to be encoded into the steering function $\mathbb{L}(\cdot)$ and this is enforced during tree additions and rewires. Any specialized sampling behavior is assumed to be part of the implementation of the subroutine `Random_Sample`.

Notes on implementation In the implementation, the heuristic measure $\mathbb{H}$ is efficiently calculated by precomputing all-pair shortest paths on the constituent graphs with *Johnson's* algorithm (Johnson 1977), which runs in $\mathcal{O}(|V|^2 log|V|)$. Precomputing the heuristic measure alleviates any overhead of spending online computation time. It is proposed that for large graphs, the vertices can be subsampled and the heuristic estimated for representative nodes that approximate the $\mathbb{H}$ value in their neighborhoods. The neighbor with the best $\mathbb{H}$ value can be computed once for a given target T, and reused during the iterations inside $\mathbb{I}_d$. In Algorithm 7 (Line 19), $\mathbb{H}$ refers to a heuristic estimate in the composite space. This can be deduced from the constituent spaces. The method also included additional focused random sampling (Gammell et al. 2015) once a solution is found to aid in convergence. For a fraction of the "random samples", goal biasing samples the target state for a robot.

The set `Adj(·)` returns the set of neighbors on the graph for a vertex, in addition to the vertex itself. This ensures that it is possible for a robot to stay static during an edge expansion. This means that the algorithm is also able to discover solutions where a subset of the robots must remain stationary for a period of time.

6 Analysis

This section examines the properties of `dRRT`* starting with the asymptotic convergence of the implicit roadmap $\hat{\mathbb{G}}$ to containing a path in $\mathbb{C}^{\text{f}}$ with optimum cost. Then, it is shown that the online search eventually discovers the shortest path in $\hat{\mathbb{G}}$. The combination of these two facts proves the asymptotic optimality of `dRRT`*and `ao-dRRT`.

For simplicity, the analysis considers robots operating in Euclidean space, i.e., $\mathbb{C}_i$ is a d-dimensional Euclidean hypercube $[0, 1]^d$ for fixed $d \geq 2$. Robots are assumed to have the same number of degrees of freedom d. The results can relate to a large class of systems, which are locally Euclidean (see, Dobson and Bekris (2013)). This is applicable to all the systems under consideration in the paper, including manipulators, with bounded angular degrees of freedom. Analysis of systems, which are not locally Euclidean, requires additional rigor especially regarding the definition of the cost metric. The Discussion section includes a description of a possible extension of the presented analysis to non-holonomic systems. It is acknowledged that the arguments presented in the current section will not readily transfer to such systems.

6.1 Optimal convergence of $\hat{\mathbb{G}}$

In this section we prove that when the connection radius $r(n)$[2] used for the construction of the single-robot `PRM` roadmaps $\mathbb{G}_1, \ldots, \mathbb{G}_R$ is chosen in a certain manner, this yields a tensor-product graph $\hat{\mathbb{G}}$, which contains asymptotically optimal paths for MMP.

Definition 3 A trajectory $\Sigma : [0, 1] \rightarrow \mathbb{C}^{\text{f}}$ is *robust* if there exists a fixed $\delta > 0$ such that for every $\tau \in [0, 1]$, $X \in \mathbb{C}^{\text{o}}$ it holds that $\|\Sigma(\tau) - X\| \geq \delta$, where $\|\cdot\|$ denotes the standard Euclidean distance.

Definition 4 Let cost be one of the cost functions defined in Sect. 3. A value $c > 0$ is *robust* if for every fixed $\epsilon > 0$,

[2] In the graphs considered here, an edge exists between two nodes, if the nodes are separated by a distance less than the connection radius $r(n)$.

there exists a robust path Σ, such that $\text{cost}(\Sigma) \leq (1+\epsilon)c$. The *robust optimum* c^*, is the infimum over all such values.

For any fixed $n \in \mathbb{N}^+$, and a specific instance of $\hat{\mathbb{G}}$ constructed from R roadmaps, having n samples each, denote by $\Sigma^{(n)}$ the lowest-cost path (with respect to $\text{cost}(\cdot)$) from S to T over $\hat{\mathbb{G}}$.

Definition 5 $\hat{\mathbb{G}}$ is asymptotically optimal (AO) if for every fixed $\epsilon > 0$ it holds that $\text{cost}(\Sigma^{(n)}) \leq (1+\epsilon)c^*$ asymptotically almost surely,[3] where the probability is over all the instantiations of $\hat{\mathbb{G}}$ with n samples for each `PRM`.

Using this definition, the following theorem is proven. Recall that d denotes the dimension of a single-robot configuration space.

Theorem 1 *$\hat{\mathbb{G}}$ is AO when*

$$r(n) \geq r^*(n) = \gamma\left(\frac{\log n}{n}\right)^{\frac{1}{d}},$$

where $\gamma = (1+\eta)2\left(\frac{1}{d}\right)^{\frac{1}{d}}\left(\frac{\mu(\mathbb{C}^{\text{f}})}{\zeta_d}\right)^{\frac{1}{d}}$ *where η is any constant larger than* 0*, μ is the volume measure and ζ_d is the volume of an unit hyperball in $\mathbb{R}^d$.*

Since the method deals with solving the problem of finding a *robust optimum* solution, some $\epsilon > 0$ is fixed. By definition of the problem, there exists a robust trajectory $\Sigma : [0,1] \rightarrow \mathbb{C}^{\text{f}}$, and a fixed $\delta > 0$, such that $\text{cost}(\Sigma) \leq (1+\epsilon)c^*$. Additionally for every $X \in \mathbb{C}^{\text{o}}$, $\tau \in [0,1]$ it holds that $\|\Sigma(\tau) - X\| \geq \delta$.

If it can be shown that $\hat{\mathbb{G}}$ contains a trajectory $\Sigma^{(n)}$, such that[4]:

$$\text{cost}(\Sigma^{(n)}) \leq (1+o(1)) \cdot \text{cost}(\Sigma) \tag{1}$$

a.a.s., this would imply that $\text{cost}(\Sigma^{(n)}) \leq (1+\epsilon)c^*$, proving Theorem 1.

As a first step, it will be shown that the robustness of $\Sigma = (\sigma_1, \ldots, \sigma_R)$ in the composite space implies robustness in the single-robot setting, i.e., robustness along σ_i.

For $\tau \in [0,1]$ define the forbidden space parameterized by τ as

$$\mathbb{C}_i^{\text{o}}(\tau) = \mathbb{C}_i^{\text{o}} \cup \bigcup_{j=1, j\neq i}^{R} I_i^j(\sigma_j(\tau)). \tag{2}$$

[3] Let $A_1, A_2 \ldots$ be random variables in some probability space and let B be an event depending on A_n. B occurs *asymptotically almost surely* (a.a.s.) if $\lim_{n\to\infty} \Pr[B(A_n)] = 1$.

[4] The small-o notation $o(1)$ indicates a function that becomes smaller than any positive constant and thereby asymptotically will become negligible. When this relation holds, the positive constant corresponds to ϵ.

Claim 1 *For every robot i, $\tau \in [0,1]$, and $q_i \in \mathbb{C}_i^{\text{o}}(\tau)$, $\|\sigma_i(\tau) - q_i\| \geq \delta$, i.e., the robustness of $\Sigma = (\sigma_1, \ldots, \sigma_R)$ in the composite space implies robustness over all single-robot paths σ_i.*

Proof Fix a robot i, and fix some $\tau \in [0,1]$ and a configuration $q_i \in \mathbb{C}_i^{\text{o}}(\tau)$. Next, define the following composite configuration

$$Q = (\sigma^1(\tau), \ldots, q_i, \ldots, \sigma^R(\tau)).$$

Note that it differs from $\Sigma(\tau)$ only in the i-th robot's configuration. By the robustness of Σ it follows that

$$\begin{aligned}\delta &\leq \|\Sigma(\tau) - Q\| \\ &= \left(\|\sigma_i(\tau) - q_i\|^2 + \sum_{j=1, j\neq i}^{R} \|\sigma_j(\tau) - \sigma_j(\tau)\|^2\right)^{\frac{1}{2}} \\ &\leq \|\sigma_i(\tau) - q_i\|.\end{aligned}$$

□

The result of Claim 1 is that the paths $\sigma_1, \ldots, \sigma_R$ are robust in their individual spaces w.r.t the parameterized forbidden space $\mathbb{C}_i^{\text{o}}(\tau)$. This means that there is sufficient clearance for the individual robots to not collide with each other given a fixed location of a single robot.

Next, a Lemma is derived using proof techniques from the literature (Janson et al. 2015, Theorem 4.1), and it implies every $\mathbb{G}_i$ contains a single-robot path $\sigma_i^{(n)}$ that converges to σ_i.

Lemma 1 *For every robot i, let $\mathbb{G}_i$ be constructed with n samples and a connection radius $r(n) \geq r^*(n)$. Then it contains a path $\sigma_i^{(n)}$ with the following attributes a.a.s.:*

(i) $\sigma_i^{(n)}(0) = s_i$, $\sigma_i^{(n)}(1) = t_i$*;*
(ii) $\|\sigma_i^{(n)}\| \leq (1+o(1))\|\sigma_i\|$*;*
(iii) $\forall q \in \text{Im}(\sigma_i^{(n)})$, $\exists \tau \in [0,1]$ *s.t.* $\|q - \sigma_i(\tau)\| \leq r^*(n)$*, where* $\text{Im}(\cdot)$ *is the function image.*

Proof The first two properties of Lemma (i) and (ii) restate (Janson et al. 2015, Theorem 4.1), which is applicable to the setup of this work. The last property (iii) is an immediate corollary of the first two: due to the fact that $\sigma_i^{(n)}$ is obtained from $\mathbb{G}_i$, every point along the path is either a vertex of the graph, or lies on a straight-line path (i.e., an edge) between two vertices, whose length is at most $r^*(n)$. □

To complete the proof of Theorem 1 it remains to be shown that the combination of $\sigma_1^{(n)}, \ldots, \sigma_R^{(n)}$ yields the trajectory, $\Sigma^{(n)}$ of a desired cost, i.e., one that conforms to Eq. 1. The bound derived in Lemma 1 (ii) looks like what we need for proving Theorem 1. Even though a similar bound exists in the individual spaces, it needs to be shown that Eq. 1 holds for a cost function in $\mathbb{C}^{\mathrm{f}}$. We proceed to show this individually for the different cost functions.

6.1.1 Optimal convergence for a linear combination of Euclidean arc lengths

Lemma 2 *Given Lemma 1* (ii)*, Eq. 1 holds for a cost function* $\mathrm{cost}(\cdot)$ *that is a linear combination of Euclidean arc lengths*

Proof Here consider the case that $\mathrm{cost}(\Sigma) = \sum_{i=1}^{R} \|\sigma_i\|$, which can also be easily modified for $\max_{i=1:R} \|\sigma_i\|$ or some arbitrary linear combination of the arc lengths.

In particular, define $\Sigma^{(n)} = (\sigma_1^{(n)}, \ldots, \sigma_R^{(n)})$, where $\sigma_i^{(n)}$ are obtained from Lemma 1. Then

$$\mathrm{cost}(\Sigma^{(n)}) = \sum_{i=1}^{R} \|\sigma_i^{(n)}\| \leq (1 + o(1)) \sum_{i=1}^{R} \|\sigma_i\| \leq (1 + o(1))\mathrm{cost}(\Sigma).$$

□

Lemma 3 *A path $\Sigma^{(n)} = (\sigma_1^{(n)} \ldots \sigma_R^{(n)})$ exists, that satisfies the properties of Lemma 1, and is collision free both in terms of robot-obstacle and robot-robot.*

Proof Every constituent roadmap $\mathbb{G}_i$ of n samples is constructed to satisfy Lemma 1 and contains individual robot paths $\sigma_i^{(n)}$. $\hat{\mathbb{G}}$ defines the tensor-product graph in the composite configuration space $\mathbb{C}$. The path $\Sigma^{(n)}$ is a combination of the individual robot paths $\sigma_i^{(n)}$. Lemma 1 implies that $\hat{\mathbb{G}}$ contains a path $\Sigma^{(n)}$ in $\mathbb{C}$, that represents collision-free motions relative to obstacles, and minimizes the cost function. Nevertheless, it is not clear whether this ensures the existence of a path where robot-robot collisions are avoided. That is, although $\mathrm{Im}(\sigma_i^{(n)}) \subset \mathbb{C}_i^{\mathrm{f}}$, it might be the case that $\mathrm{Im}(\Sigma^{(n)}) \cap \mathbb{C}^{\mathrm{o}} \neq \emptyset$. Next, it is shown that $\sigma_1^{(n)}, \ldots, \sigma_R^{(n)}$ can be reparametrized to induce a composite-space path whose image is fully contained in $\mathbb{C}^{\mathrm{f}}$, with length equivalent to $\Sigma^{(n)}$.

For each robot i, denote by $V_i = (v_i^1, \ldots, v_i^{\ell_i})$ the chain of $\mathbb{G}_i$ vertices traversed by $\sigma_i^{(n)}$. For every $v_i^j \in V_i$ assign a timestamp τ_i^j of the closest configuration along σ_i, i.e.,

$$\tau_i^j = \underset{\tau \in [0,1]}{\mathrm{argmin}} \|v_i^j - \sigma_i(\tau)\|.$$

Also, define $\mathcal{T}_i = (\tau_i^1, \ldots, \tau_i^{\ell_i})$ and denote by $\mathcal{T}$ the ordered list of $\bigcup_{i=1}^{R} \mathcal{T}_i$, according to the timestamp values. Now, for every i, define a global timestamp function $TS_i : \mathcal{T} \to V_i$, which assigns to each global timestamp in $\mathcal{T}$ a single-robot configuration from V_i. It thus specifies in which vertex robot i resides at time $\tau \in \mathcal{T}$. For $\tau \in \mathcal{T}$, let j be the largest index, such that $\tau_i^j \leq \tau$. Then simply assign $TS_i(\tau) = \tau_i^j$. From property (iii) in Lemma 1 and Claim 1 it follows that no robot-robot collisions are induced by the reparametrization. This concludes the proof of Theorem 1. □

6.1.2 Optimal convergence for Euclidean arc length

Arguments for convergence of the cost of the solution in terms of the Euclidean arc length of the composite path Σ can be made to extend the results of Lemma 2. A robot having d DoFs, $(F_1, \ldots F_d)$ exists in an $\mathbb{R}^d$ space. The motion of the robot constitutes a curve in $\mathbb{R}^d$, defined as

$$\Sigma = (f_1(t), \ldots f_d(t)),$$

where $f_i(t)$ is the coordinate function of the curve Σ along the DoF F_i, where $t \in [0, 1]$ and $i \in [1, d]$. The *Euclidean* arc-length of the path in $\mathbb{R}^d$:

$$\|\Sigma\| = \int \sqrt{(f_1'(t)^2 + \cdots + f_d'(t)^2)}\, dt. \tag{3}$$

The coordinate function is assumed to be continuous with respect to the Lebesgue measure on t and of bounded variation. The Lebesgue measure assigns a measure to subsets of n-dimensional Euclidean space. For $n = 1, 2$, or 3, it coincides with the standard measure of length, area, or volume. The assumption here is that the curve is Lebesgue integrable over the coordinate function. This relates to the variation of the curve being smooth for the parametrization, over subsets of the curve that correspond to the Lebesgue measure over t. Σ is also a rectifiable curve, i.e., the curve has finite length.

Definition 6 The partial arc length is defined as,

$$s(x) = \|\Sigma : [0, x]\|, x \leq 1.$$

By definition and assuming smoothness and bounded variation (Pelling 1977), for $t = 0$ to $t = x \leq 1$:

$$s(x) = \sup_P \sum_{j=1}^{m} \sqrt{\left(\sum_{i=1}^{d} (f_i(t_j) - f_i(t_{j-1}))^2\right)}. \tag{4}$$

The value is the supremum over all possible finite partitions $P : 0 = t_0 < \cdots < t_m = x$, that can divide t. This generates a finite set of m parts.

We denote the value of $s(x)$ for some partition P as:

$$s(x)_P = \sum_{j=1}^{m} \sqrt{\sum_{i=1}^{d} (f_i(P(j)) - f_i(P(j-1)))^2}. \tag{5}$$

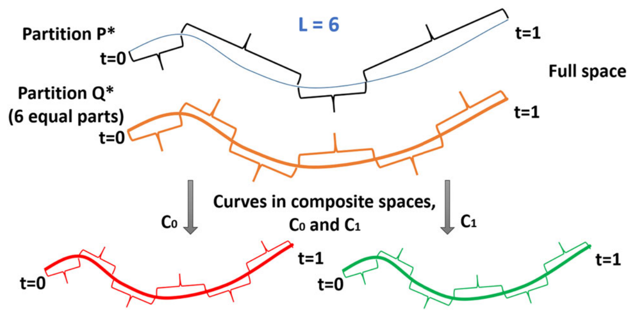


Fig. 6 Partitions of the curves in the full space and the composite spaces (bottom). P^* is the supremum partition but the parts do not have equal measures. P^* is subdivided into a finer partition Q with $L = 6$ equal measure parts. Let the composite curves exist in individual robot configuration spaces $\mathbb{C}_0$ and $\mathbb{C}_1$, respectively. These curves also have a finer partition with 6 equal parts

A *part* shall refer to the curve between $\Sigma(P(j-1))$ and $\Sigma(P(j))$. The measure of the part would be

$$\mu^P_{j-1,j} = s(P(j)) - s(P(j-1)). \tag{6}$$

Let P^* be the *finite* supremum partitioning over t, that has m parts. This means, $s(x) = s(x)_{P^*}$.

Without loss of generality, let us assume that P^* corresponds to the supremum partitioning that has the least number of parts, $|P^*| = m+1$, i.e., there are no degenerate partitions. A finer partition can introduce additional parameterization over t, and hence is a superset of P^*, but cannot increase the value of $s(x)$ since P^* is the supremum. Note that a partition sequence with m parts will have a cardinality of $m+1$. The finer partition has all these $m+1$ parametrization values in addition to others. Since $s(x)$ is finite and m is finite, $\exists P^* \mid \mu^{P^*}_{j-1,j} \in \mathbb{R}_+ \; \forall\, j \in t$.

Claim 2 *Given a finite set of paths ξ, there exists a finer partitioning, Q^* for P^* over each $\Sigma \in \xi$, which yields L number of equal measure parts (Fig. 6) for every $\Sigma \in \xi$.*

Proof

$$\exists L \text{ s.t. } s(1)_{Q^*} = s(1)_{P^*} = \|\Sigma\|,$$
$$Q^* \supseteq P^*,\; |Q^*| = L+1,$$
$$\mu_{j-1,j} = \mu_{k-1,k} = \frac{\|\Sigma\|}{L} \in \mathbb{R}_+ \;\; \forall\, j,k \in [1,\ldots L].$$

This holds true for every Σ for a corresponding $Q^* \supseteq P^*$. The measure of every *part* $\Sigma(l)$ is equal, and is denoted by $\|\Sigma(l)\| = \frac{\|\Sigma\|}{L}$. This simplifies Eq. 5 to $\|\Sigma\| = \sum_{l=1}^{L} \|\Sigma(l)\|$. □

Claim 3 *Additionally, by this simplification, Eq. 5 in the composite space is restated for $\Sigma_{Rd} = \{\Sigma_1 \ldots \Sigma_R\}$ where $\Sigma_{Rd}, \Sigma_1 \ldots \Sigma_R \in \xi$.*

Proof In the multi-robot space Euclidean space $\mathbb{R}^{Rd}$, the arc length in the composite space can be expressed in terms of the arc lengths traversed in the individual robot spaces.

$$\begin{aligned}\|\Sigma_{Rd}\| &= s(1)_{Q^*}\\ &= \sum_{l=1}^{L}\sqrt{\sum_{i=1}^{Rd}(f_i(Q^*(j)) - f_i(Q^*(j-1)))^2}\\ &= \sum_{l=1}^{L}\sqrt{\sum_{i=1}^{d}(\delta f_i)^2 + \cdots + \sum_{i=(R-1)d+1}^{Rd}(\delta f_i)^2}\\ &= \sum_{l=1}^{L}\sqrt{\|\Sigma_1(l)\|^2 + \cdots + \|\Sigma_R(l)\|^2}\\ &= \sum_{l=1}^{L}\sqrt{\sum_{i=1}^{R}\|\Sigma_i(l)\|^2},\end{aligned}$$

where, with a slight abuse of notation δf_i is a shorthand representation for some $f_i(Q^*(j)) - f_i(Q^*(j-1))$. □

Lemma 4 *For a $\Sigma^{(n)} = (\sigma_1^{(n)} \ldots \sigma_R^{(n)})$, where $\sigma_i^{(n)}$ is obtained from Lemma 1, given that $\|\sigma_i^{(n)}\| \le (1+o(1))\|\sigma_i\|$, Eq. 1 holds for the Euclidean arc lengths.*

Proof Partitioning the arcs $\sigma_i^{(n)}$, and σ_i, into L (chosen as per Claim 2) pieces of equal length, yields two trajectory sequences, for $l \in N_+, l \le L$.

The high level idea is that leveraging the uniformity in the parameterized *parts* introduced by L, Lemma 1 (ii) has to be recombined to represent the Euclidean arc length in the composite space.

$$\|\sigma_i^{(n)}\| \le (1+o(1))\|\sigma_i\| \qquad \text{(Using Lemma 1)}$$

$$\Rightarrow \sum_{l=1}^{L}\|\sigma_i^{(n)}(l)\| \le (1+o(1))\sum_{l=1}^{L}\|\sigma_i(l)\|$$

(Using Claim 2 and 3)

$$\Rightarrow \|\sigma_i^{(n)}(l)\| \le (1+o(1))\|\sigma_i(l)\|$$
$$\Rightarrow \|\sigma_i^{(n)}(l)\|^2 \le (1+o(1))^2\|\sigma_i(l)\|^2.$$

Combining over R using Claim 3,

$$\sum_{i=1}^{R}\|\sigma_i^{(n)}(l)\|^2 \le (1+o(1))^2\sum_{i=1}^{R}\|\sigma_i(l)\|^2$$

$$\Rightarrow L\sqrt{\sum_{i=1}^{R}\|\sigma_i^{(n)}(l)\|^2} \le (1+o(1))L\sqrt{\sum_{i=1}^{R}\|\sigma_i(l)\|^2}$$

$$\Rightarrow \|\Sigma^{(n)}\| \le (1+o(1))\|\Sigma\|$$

□

Following the same parameterization described in Lemma 3, Theorem 1 can be shown for the Euclidean metric as well.

6.2 Asymptotic optimality of dRRT*

Finally, dRRT* is shown to be AO. Denote by m the time budget in Algorithm 6, i.e., the number of iterations of the loop. Denote by $\Sigma^{(n,m)}$ the solution returned by dRRT* for n samples in the individual constituent roadmaps and m iterations of the dRRT* algorithm.

Theorem 2 *If $r(n) > r^*(n)$ then for every fixed $\epsilon > 0$ it holds that*

$$\lim_{n,m\to\infty} \Pr\left[\mathrm{cost}(\Sigma^{(n,m)}) \leq (1+\epsilon)c^*\right] = 1.$$

Since $\hat{\mathbb{G}}$ is AO (Theorem 1), it suffices to show that for any fixed n, and a fixed instance of $\hat{\mathbb{G}}$, defined over R PRMs with n samples each, dRRT* eventually (as m tends to infinity), finds the optimal trajectory over $\hat{\mathbb{G}}$. This property is stated in Lemma 5 and proven subsequently. The same arguments hold for both dRRT* and ao-dRRT, with the difference highlighted explicitly in the proof.

Lemma 5 (Optimal Tree Convergence of dRRT*) *Consider an arbitrary optimal path π^* originating from v_0 and ending at v_t, then let $O_k^{(m)}$ be the event such that after m iterations of* dRRT*, *the search tree $\mathbb{T}$ contains the optimal path up to segment k. Then,*

$$\liminf_{m\to\infty} \mathbb{P}\big(O_t^{(m)}\big) = 1.$$

Proof This is shown using Markov chain results (Grinstead and Snell 2012, Theorem 11.3). Specifically, absorbing Markov chains can be leveraged to show that dRRT* will eventually contain the optimal path over $\hat{\mathbb{G}}$. An absorbing Markov chain has some subset of its states in which the transition matrix only allows self-transitions.

The proof follows by showing that the dRRT* method can be described as an absorbing Markov chain, where the target state of a query is represented as an absorbing state in a Markov chain. For completeness, the theorem is re-stated here.

Theorem 3 (Thm 11.3 in Grinstead & Snell) *In an absorbing Markov chain, the probability that the process will be absorbed is 1 (i.e., $Q(m) \to 0$ as $n \to \infty$), where $Q(m)$ is the transition submatrix for all non-absorbing states.*

The first part is that the dRRT* search is cast as an absorbing Markov chain, and second, that the transition probability from each state to the next is nonzero, i.e., each state eventually connects to the target.

For query (S, T), let the sequence $V = \{v_1, v_2, \cdots, v_t\}$ of length t represent the vertices of $\hat{\mathbb{G}}$ corresponding to the optimal path through the graph which connects these points, where v_t corresponds to the target vertex, and furthermore, let v_t be an absorbing state. Theorem 3 works under the assumption that each vertex v_i is connected to an absorbing state v_t.

Then, let the transition probability for each state have two values, one for each state transitioning to itself, which corresponds to the dRRT* search expanding along some other arbitrary path. The other value is a transition probability from v_i to v_{i+1}. This corresponds to two slightly different cases for ao-dRRT and dRRT*.

Case ao-dRRT The transition probability from v_i to v_{i+1} corresponds to the method sampling within the volume $\mathrm{Vol}(v_i)$. Then, as the second step, it must be shown that this volume has a positive probability of being sampled in each iteration. It is sufficient then to argue that $\frac{\mu(\mathrm{Vol}(s_i))}{\mu(\mathbb{C}^f)} > 0$. Fortunately, for any finite n, previous work has already shown that this is the case given general position assumptions (Solovey et al. 2015a, Lemma 2).

Case dRRT* In the case of dRRT* due to the random neighborhood selection in the expansion $\mathbb{I}_d$, there is a positive transition probability from v_i to v_{i+1}.

Given these results, the dRRT* is cast as an absorbing Markov chain, which satisfies the assumptions of Theorem 3, and therefore, the matrix $Q(m) \to 0$. This implies that the optimal path to the goal has been expanded in the tree, and therefore $\liminf_{m\to\infty} \mathbb{P}\big(O_t^{(m)}\big) = 1$. □

7 Extension to shared degrees of freedom

This section describes an extension of the dRRT* approach to systems with shared degrees of freedom (DoF), with specific focus on humanoid robots with two arms. The challenge here arises because of the high dimensionality of the robots. The shared DoF is a general formulation, which can refer to either degrees of freedom in a torso or a mobile base etc.

This section is structured in the same way as the rest of the algorithmic descriptions, and a lot of the shared notations and details are omitted for the sake of brevity. Instead, the interesting insights into the problems that arise due to the shared DoF are highlighted, and resolved. A high level overview of the differences of dual-arm dRRT* (da-dRRT*) from the previously stated methods includes:

- da-dRRT* decomposes the space by grouping the shared DoF with one of the arms.
- da-dRRT* implicitly builds two trees online that explores two tensor roadmaps.

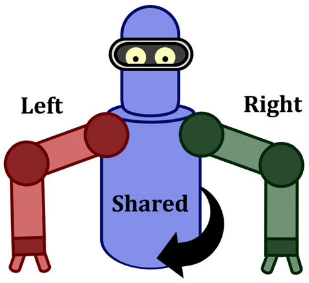


Fig. 7 The decomposition of the full configuration space into the arm spaces ($\mathbb{C}_l$ and $\mathbb{C}_r$), and the shared degrees of freedom ($\mathbb{C}_s$) such that $\mathbb{C} = \mathbb{C}_l \times \mathbb{C}_s \times \mathbb{C}_r$

– da-dRRT* needs additional arguments for proving robustness in Claim 1.

The current work does not get into aspects related to manipulation. Nevertheless, the primitives designed here can speed up dual-arm manipulation task planning, where computational benefits can be achieved by operating over multiple roadmaps (Gravot et al. 2002; Gravot and Alami 2003b). The topology of dual-arm manipulation has been formalized (Koga and Latombe 1994; Harada et al. 2014) and extended to the N-arm case (Dobson and Bekris 2015). It requires the consideration of multi-robot grasp planning (Vahrenkamp et al. 2010; Dogar et al. 2015), regrasping (Vahrenkamp et al. 2009), as well as closed kinematic chain constraints (Cortés and Siméon 2005; Bonilla et al. 2017). Furthermore, force control strategies are helpful for multi-arm manipulation of a common object (Caccavale and Uchiyama 2008). Recently coordinated control has been applied to solve human-robot interaction tasks (Sina Mirrazavi Salehian et al. 2016).

The algorithm is meant to address the applicability of dRRT* to high dimensional humanoid robots with shared DoF.

7.1 Problem setup and notation

As shown in Fig. 7, the DoF$[F_1, \ldots, F_d]$ can be grouped into left, right, and shared DoF subsets, so that: $\mathbb{C} = \mathbb{C}_l \times \mathbb{C}_s \times \mathbb{C}_r$. A candidate solution path $\Sigma : [0, 1] \rightarrow \mathbb{C}^f$ can be decomposed to projections $[\Sigma_l, \Sigma_s, \Sigma_r]$ along $\mathbb{C}_l$, $\mathbb{C}_s$ and $\mathbb{C}_r$ respectively.

The method proposes the construction of the following roadmaps, as shown in Fig. 8:

– A left-shared $\mathbb{R}_{ls}(\mathbb{V}_{ls}, \mathbb{E}_{ls})$ and a right-shared DoF roadmap $\mathbb{R}_{sr}(\mathbb{V}_{sr}, \mathbb{E}_{sr})$, where $\mathbb{V}_{ls} \subset \mathbb{C}_l \times \mathbb{C}_s$ and $\mathbb{V}_{sr} \subset \mathbb{C}_s \times \mathbb{C}_r$. The edges are collision-free paths in the same spaces, i.e., no collisions with the static geometry, or self-collisions among the arm or the shared DoFs.
– A left arm $\mathbb{P}_l(\mathbb{V}_l, \mathbb{E}_l)$ and a right arm roadmap $\mathbb{P}_r(\mathbb{V}_r, \mathbb{E}_r)$, such that $\mathbb{V}_l \subset \mathbb{C}_l$, and $\mathbb{V}_r \subset \mathbb{C}_r$. These roadmaps do not consider the static geometry as they are not grounded by the shared DoFs. So, only self-collisions between arm links are avoided.

The method focuses on two *tensor product roadmaps*: $\hat{\mathbb{G}}_l = \mathbb{R}_{ls} \times \mathbb{P}_r$, and $\hat{\mathbb{G}}_r = \mathbb{R}_{sr} \times \mathbb{P}_l$. The method then simultaneously searches over $\hat{\mathbb{G}}_l$ and $\hat{\mathbb{G}}_r$ in a dRRT*-esque fashion.

7.2 Methodology

This section describes the proposed method, and the way the da-dRRT* builds a forest of two trees $\mathbb{T}$, which explores both $\hat{\mathbb{G}}_l$ and $\hat{\mathbb{G}}_r$. In terms of the method's properties it is sufficient to consider only one roadmap, but in practice, exploring them simultaneously helps in the convergence, since we can evaluate more possible solutions and rewires. The approach shows faster convergence compared to RRT* in $\mathbb{C}$, and scales more than PRM*.

At a high-level, the proposed Dual-arm dRRT* (da-dRRT*) simultaneously explores the tensor product roadmaps $\hat{\mathbb{G}}_l$ and $\hat{\mathbb{G}}_r$, by building a search tree for each one so as to find a solution from the start configuration S to the target configuration T. For every vertex, the algorithm keeps track from which *tensor product roadmap* the vertex belongs to. Upon initialization, the tree starts with two vertices, S_l and S_r, one corresponding to tensor product roadmap $\hat{\mathbb{G}}_l$ and the other to $\hat{\mathbb{G}}_r$. Then, at every iteration, the tree data structure $\mathbb{T}$ is expanded by adding a new edge and a node by calling an expand subroutine like Algorithm 7. The differences arises in the neighborhood calculation in Algorithm 7 Line 8. The neighborhood N for V^{new} considers the tensor roadmap neighborhoods that are part of the tree for both roadmaps. V^{new} belongs to to either $\hat{\mathbb{G}}_l$ or $\hat{\mathbb{G}}_r$. $V^{\hat{\text{new}}}$ is chosen to be the nearest tree vertex that was generated on the other tensor roadmap. N is the set of all tree vertices that are tensor roadmap neighbors of V^{new} or $V^{\hat{\text{new}}}$. While doing rewires, care is taken to only rewire nodes belonging to the same tensor roadmap. The consideration of a richer neighborhood lets the algorithm ensure adequate exploration of both tensor roadmaps. The informed oracle $\mathbb{I}_d$ is similar to Algorithm 8 with the difference arising for the constituent roadmap $\mathbb{P}$, where the $\mathbb{H}$ estimate is simply the shortest Euclidean distance to the goal.

Notes on efficiency The difference of the decomposition for shared degrees of freedom compared to dRRT* is that $\mathbb{G} = \mathbb{R} \times \mathbb{P}$ does not give two kinematically independent spaces. Specifically, $\mathbb{P}$ depends on the shared DoF to be grounded to the frame of the robot. This means that the heuristic $\mathbb{H}$ is less informed for $\mathbb{P}$ and can only use the straight line distance. The dRRT* algorithm does not work out of the box

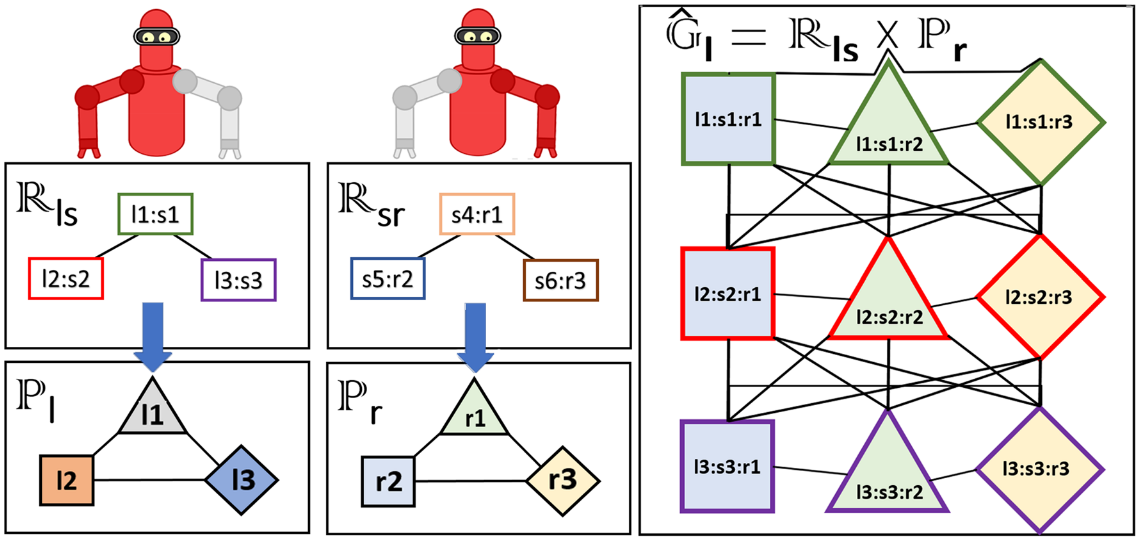


Fig. 8 The image on the left is an illustration of the decomposition of the space to create arm-shared DoF roadmaps $\mathbb{R}$ and arm only roadmaps $\mathbb{P}$. The example has three vertices in each roadmap consisting of a combination Left (l), Shared (s), and Right(r) values. For the sake of clarity the vertices on the arm-only roadmaps correspond to the $\mathbb{R}$ roadmaps. The image on the right shows the connectivity in the tensor product roadmap $\hat{\mathbb{G}}_l = \mathbb{R}_{ls} \times \mathbb{P}_r$. A similar tensor product is obtained for $\hat{\mathbb{G}}_r$

in the case of robots with shared degrees of freedom. The effect of the less expressive heuristic in da-dRRT*, translates into some degradation in performance relative to the case of two kinematically independent robotic arms. Nevertheless, da-dRRT* is still significantly faster than operating directly in the composite space of the entire robot. There are not many methods that can practically compute solutions for such high-dimensional (e.g., 15 degrees of freedom) systems with kinematic dependences. The proposed da-dRRT* method preserves some of the scalability benefits of da-dRRT*and addresses the kinematic dependence that arises for many popular humanoid robots.

7.3 Analysis

Asymptotic optimality of tensor roadmaps Given the decomposition, $\mathbb{C}$ is divided into two parts: $\mathbb{R}_{ls}$ and $\mathbb{P}_r$.

If a robust optimal path Σ^* exists in $\mathbb{C}$, most of the arguments of Sect. 6 still hold for this decomposition. Due to the nature of the space decomposition, since the constituent spaces do not correspond to kinematically independent robots, the clearance assumption in Claim 1 needs to be reworked.

Claim 4 *Robustness in $\mathbb{C}$ implies robustness in $\mathbb{C}_{ls}$ and $\mathbb{C}_r$. For every decomposition, $\tau \in [0, 1]$, and $q_i \in \mathbb{C}_i^o(\tau)$, $\|\sigma_i(\tau) - q_i\|_2 \geq \delta$.*

Proof Consider any $Q = (\Sigma_{ls}(\tau), q_r)$, where q_r is a configuration in $\mathbb{C}_r$ so that the right arm collides either with the static geometry or with the left-shared part of the robot, which is at $\sigma_{ls}(\tau)$. Given a robust Σ, Q is a colliding configuration: $\delta \leq \|\Sigma(\tau) - Q\|$. But Q and $\Sigma(\tau)$ only differ in q_r, so the path σ_r has clearance δ

$$\delta \leq ||\sigma_r(\tau) - q_r||.$$

By switching the decomposition of Σ in $\hat{\mathbb{G}}_l$ into (σ_l, σ_{sr}), by the above reasoning:

$$\delta \leq ||\Sigma(\tau) - Q|| \implies \delta \leq ||\Sigma_l(\tau) - q_l||$$

Now, since $\forall \tau : \quad ||\sigma_{ls}(\tau) - q_{ls}|| \geq ||\sigma_l(\tau) - q_l||$
$$\implies \delta \leq ||\sigma_{ls}(\tau) - q_{ls}||.$$

This proves the robustness for σ_{ls}. The same reasoning can be applied to $\mathbb{C}_{sr}$ and $\mathbb{C}_l$. □

It suffices to follow the proof structures outlined in Sect. 6 to argue asymptotic optimality for the method. It should be noted that due to the coupled nature of $\mathbb{C}$ introduced by the shared DoF, the use of the Euclidean cost metric is more applicable.

8 Experimental validation

This section provides an experimental evaluation of dRRT* by demonstrating practical convergence, scalability for disk robots, and applicability to dual-arm manipulation. The choice of a cost metric depends on the type of application and the underlying system properties. For systems without shared degrees of freedom, the considered cost function is the sum of individual Euclidean arc lengths, which is a popular choice for multi-robot systems. For systems with shared degrees of freedom, the combined nature of the underlying configuration space motivates the use of Euclidean arc length in the composite space as the metric. The results show that the properties and benefits of the proposed algorithms stay robust for both choices of cost functions.

8.1 Tests on systems without shared DoF

The approach and alternatives are executed on a cluster with Intel(R) Xeon(R) CPU E5-4650 @ 2.70 GHz processors, and 128 GB of RAM. The solution costs are evaluated in terms of the sum of Euclidean arc lengths.

2 disk robots among 2D polygons This base-case test involves 2 disks ($\mathbb{C}_i := \mathbb{R}^2$) of radius 0.2 with bounded velocity, in a 10×10 region, inflated by the radius, as in Fig. 9. The disks have to swap positions between (0, 0) and (9, 9). This is a setup where it is possible to compute the explicit roadmap, which is not practical in more involved scenarios. In particular, dRRT* is tested against: (a) running A* on the implicit tensor roadmap $\hat{\mathbb{G}}$ (referred to as "Implicit A*"), where $\hat{\mathbb{G}}$ is defined over the same individual roadmaps with N nodes as those used by dRRT*; (b) an explicitly constructed PRM* roadmap with N^2 nodes in $\mathbb{C}$; and (c) the ao-dRRT variant of the algorithm.

Results are shown in Fig. 10. dRRT* converges to the optimal path over $\hat{\mathbb{G}}$, similar to the one discovered by Implicit A*, while quickly finding an initial solution of high quality. Furthermore, the implicit tensor product roadmap $\hat{\mathbb{G}}$ is of comparable quality to the explicitly constructed roadmap. The convergence of dRRT* is faster compared to corresponding ao-dRRT variant as evident from Fig. 10(left).

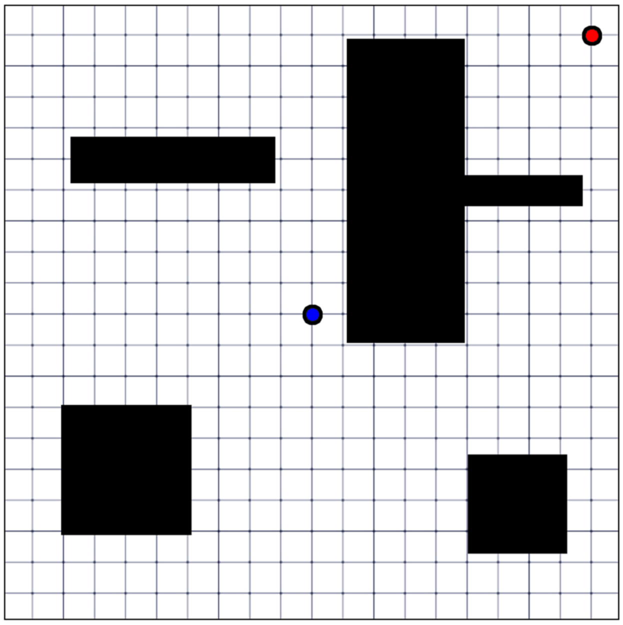


Fig. 9 The 2D environment where the 2 disk robots operate

Table 1 presents running times. dRRT* and implicit A* construct 2 N-sized roadmaps (row 3), which are faster to construct than the PRM* roadmap in $\mathbb{C}$ (row 1). PRM* becomes very costly as N increases. For $N = 500$, the explicit roadmap contains 250,000 vertices, taking 1.7 GB of RAM to store, which was the upper limit for the machine used. When the roadmap can be constructed, it is fast to query (row 2). dRRT* quickly returns an initial solution (row 5), at par with the solution times from the explicit roadmap and well before Implicit A* returns a solution (row 4). The initial solution times are compared visually in Fig. 10 which demonstrates the efficiency of dRRT* compared to ao-dRRT as well. The next benchmark further emphasizes this point.

The comparison between the early solution time required to find a suboptimal solution by the proposed method against the computation time needed by the optimal A* highlights the impact of roadmaps of increasing sizes. While dRRT*'s initial solution times barely change, the time taken by any variant of heuristic search over the composite roadmap increases with the size of the roadmap. This indicates that roadmaps of size similar to the tensor roadmaps considered here would rapidly cease to be solvable without anytime performance similar to that of dRRT*.

Many disk robots among 2D polygons In the same environment as above, the number of robots R is increased to evaluate scalability. The same environment is maintained in this benchmark to introduce additional complexity purely in terms of the addition of more robots into the planning problem. The effect of more difficult and practical planning scenarios would be explored in the subsequent benchmarks with manipulators. Each robot starts on the perimeter of the environment and is tasked with reaching the opposite side. An $N = 50$ roadmap is constructed for every robot. It quickly becomes intractable to construct a PRM* roadmap in the composite space of many robots.

Figure 11 shows the inability of alternatives to compete with dRRT* in scalability. Solution costs are normalized by an optimistic estimate of the path cost for each case, which is the sum of the optimal solutions for each robot, disregarding robot-robot interactions. The colored vertical bars represent the range of the average initial and final solution costs. Implicit A* fails to return solutions even for 3 robots. Directly executing RRT* in the composite space fails to do so for $R \geq 6$. The original dRRT method (without the informed search component) starts suffering in success ratio for $R \geq 4$ and returns worse quality solutions than dRRT*. The ao-dRRT variant performs similar to dRRT in terms of

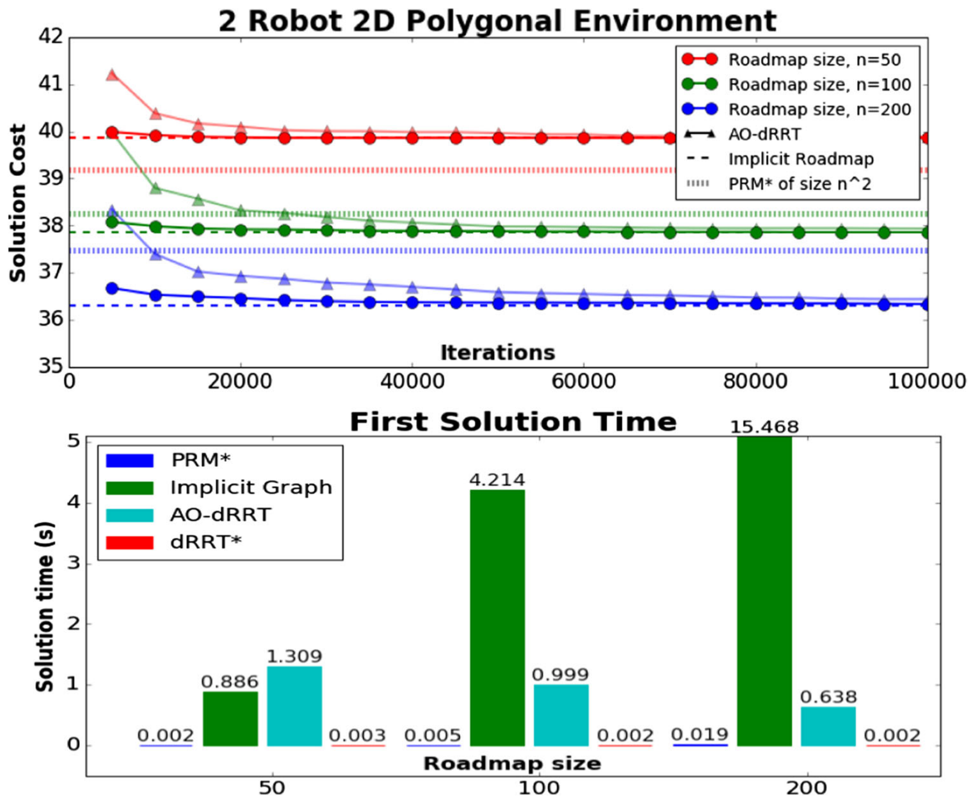


Fig. 10 For every n, 10 randomly generated pairs of roadmaps are generated. dRRT* and ao-dRRT ran on 5 random experiments for every roadmap pair, and the implicit A^* searches these 10 tensor combinations. PRM* is run 10 times for every n. *(Top)*: Average solution cost is reported over iterations. Data averaged over 10 roadmap pairs. dRRT* (solid circled line) and ao-dRRT (solid triangled line) converges to the optimal path through $\hat{\mathbb{G}}$ (dashed line). *(Bottom)*: Initial solution times for the algorithms

Table 1 Construction and query times (s) for 2 disk robots

Number of nodes: $N =$	50	100	200
N^2-PRM* construction	3.427	13.293	69.551
N^2-PRM* query	0.002	0.005	0.019
2 N-size PRM* construction	0.135	0.274	0.558
Implicit A* search over $\hat{\mathbb{G}}$	0.886	4.214	15.468
ao-dRRT over $\hat{\mathbb{G}}$ (initial)	1.309	0.999	0.638
dRRT* over $\hat{\mathbb{G}}$ (initial)	0.003	0.002	0.002

success ratio but expectedly finds better solutions than dRRT. dRRT* finds solutions up to $R = 10$.

In order to give an estimate of the immensity of the size of the search space, for $R = 10$, the *tensor-product roadmap* represents an implicit structure consisting of 50^{10} or ~ 100 million-billion vertices.

Dual-arm manipulator This test (Fig. 12) shows the benefits of dRRT* when planning for two 7-dimensional arms. Figure 13 shows that RRT^* fails to return solutions within $100K$ iterations. Using small roadmaps is also insufficient for this problem. Both dRRT* and Implicit A^* require larger roadmaps to begin succeeding. But with $N \geq 500$, Implicit A^* always fails, while dRRT* maintains a 100% success ratio. As expected, roadmaps of increasing size result in higher quality path. The informed nature of dRRT* also allows to find initial solutions fast, which together with the branch-and-bound primitive allows for good convergence. The initial solution times in Fig. 13 indicate that the heuristic guidance succeeds in finding fast initial solutions even for larger roadmaps.

8.2 Tests on systems with shared DoF

This section showcases three benchmarks of increasing difficulty, which are used to evaluate the performance of the da-dRRT*. All the experiments were run on a cluster with Intel(R) Xeon(R) CPU E5-4650 @ 2.70 GHz processors, and 128 GB of RAM. In each benchmark, different sizes n of the constituent roadmaps $\mathbb{R}_{ls}$ and $\mathbb{R}_{sr}$ were evaluated. The da-dRRT*algorithm is compared against $\text{RRT}^*$and PRM*. The platforms used are Motoman SDA10F, with a torsional DoF, and Baxter on a mobile base that can rotate and translate. For the PRM* algorithm and all benchmarks, 20 randomly seeded roadmaps with 50,000 nodes are constructed in $\mathbb{C}$ and data are gathered from 20 experiments. A 50,000 node roadmap has ≈ 1 million edges, and takes ≈ 7 h to construct in these high dimensional spaces. Larger roadmaps run into memory scalability issues. These roadmaps in the full space occupied ≈ 50 MB. In comparison, the space requirement for two arm roadmaps were < 1 MB.

For all benchmarks, both RRT^* and da-dRRT*were allowed to run for 100,000 iterations. RRT^* is ran in 20

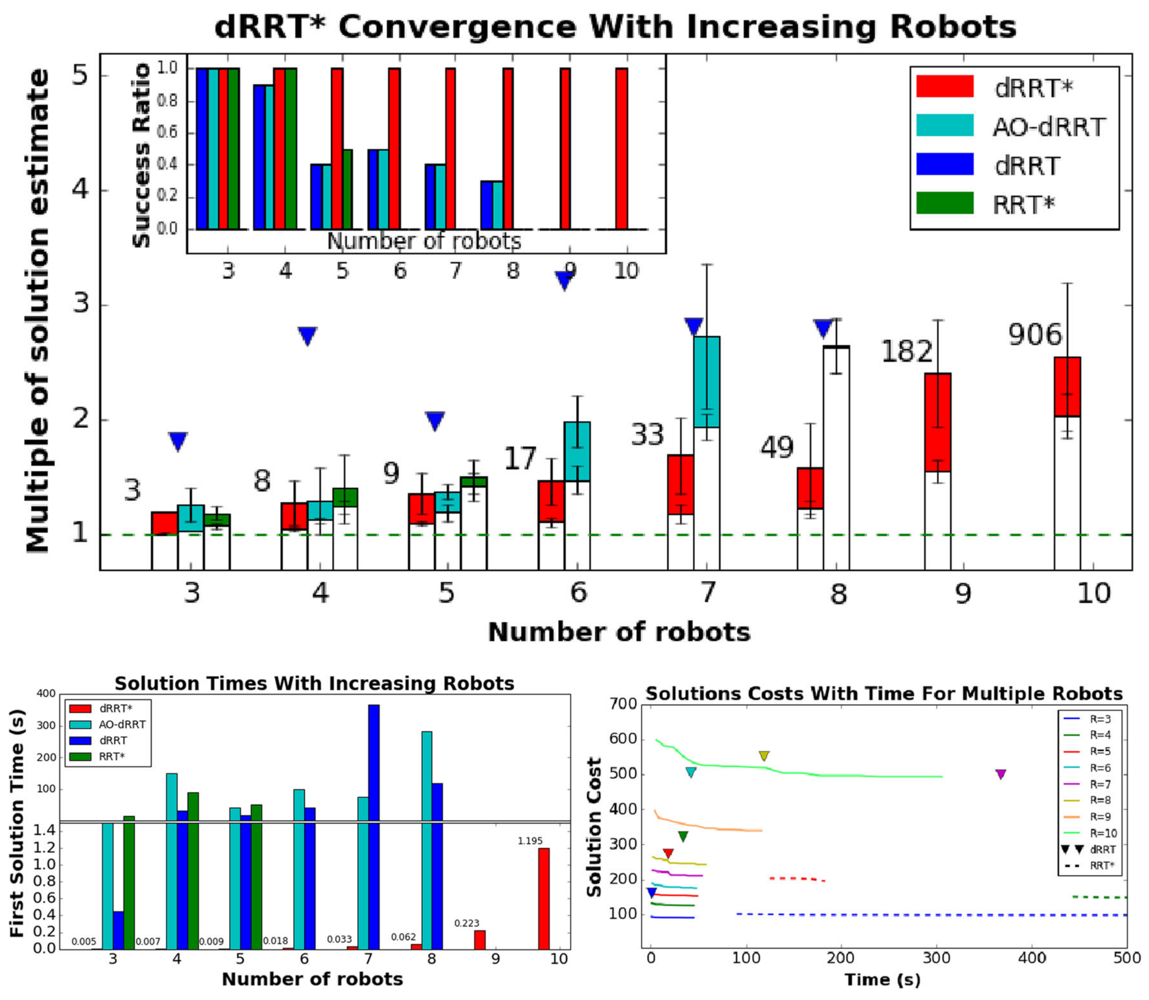


Fig. 11 Data averaged over 10 runs for $R = 3$ to 10 robots. The data is reported for the algorithms dRRT*, ao-dRRT, dRRT, and RRT*. (*Top inset*): The success ratio shows the fraction of the runs that returned a solution. (*Top*): Relative solution cost of the algorithms for increasing R over 100,000 iterations. The horizontal green line at 1 denotes the best possible cost estimate, which is a combination of the individual robot shortest paths for each problem. All the other costs are represented as multiples of this estimate. dRRT only reports a single solution, and is denoted by the inverted blue triangles. The other algorithms improve the solution over the iterations, represented by vertical bars between the average initial solution cost and average final reported solution cost. The numbers above the dRRT* bars represent the iteration number of the first solution for dRRT*. (*Bottom left*): The average initial solution times for the algorithms. (*Bottom right*): The plot of the reported solution cost over time for the different algorithms. dRRT only reports a single solution and is represented by the inverted triangles

different randomly seeded experiments for every benchmark. For the da-dRRT* algorithm, 20 experiments are run for every benchmark, for the different constituent roadmap sizes n, by building 4 pairs of randomly seeded constituent roadmaps, and running 5 randomly seeded experiments over each roadmap combination.

Motoman tabletop benchmark A set of 20 random collision-free starts and goals are selected in the tabletop environment, shown in Fig. 14.

They are only used if they are sufficiently far away from each other. da-dRRT*is tested with constituent roadmap sizes of 100, 250 and 500. All the algorithms succeed in every experiment. In this simpler problem, smaller roadmaps are quicker to search, and generate initial solutions faster compared to RRT*, as shown in Fig. 15 (*top*).

Searching the PRM* is the fastest (online), but the solution quality is worse than that obtained from the other methods. da-dRRT* converges to better solutions, compared to the other algorithms, as shown in Fig. 15 (*bottom*).

Motoman shelf benchmark This benchmark sets up the Motoman in front of 3 shelves. The robot has to plan between two states where both arms are inside different shelving units, which require the rotation of its torso (Fig. 16 (*top*)).

This is a significantly harder problem, and RRT* suffers in terms of success ratio (Fig. 16 (*second*)). RRT* takes much longer to find the initial solution, as indicated by Fig. 16

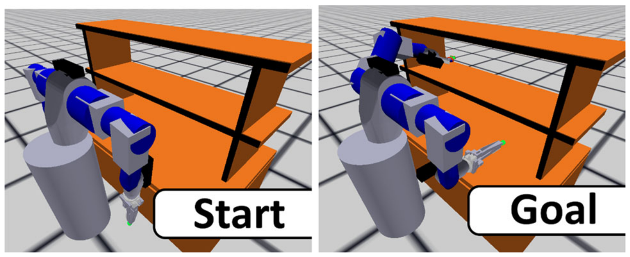


Fig. 12 The start and target configuration for the dual-arm manipulator benchmark on a `Motoman SDA10F`. `dRRT`* is run for a dual-arm manipulator to go from its home position (left) to a reaching configuration (right)

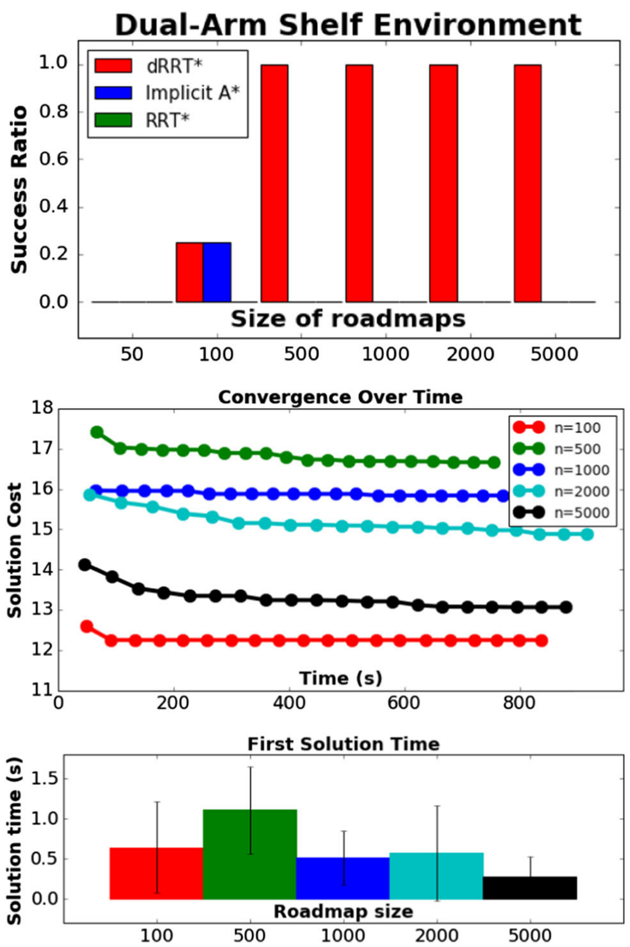


Fig. 13 (*Top*): 5 random experiments are run for 4 random roadmap pairs for every n. `dRRT`* achieves perfect success ratio as n increases. (*Middle*): `dRRT`* solution quality over time. Here, larger roadmaps provide benefits in terms of running time and solution quality. (*Bottom*): Initial solution times for `dRRT`*

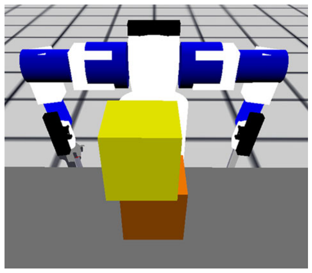


Fig. 14 The Motoman tabletop benchmark setup for `da-dRRT`* with randomly sampled start and goal configurations

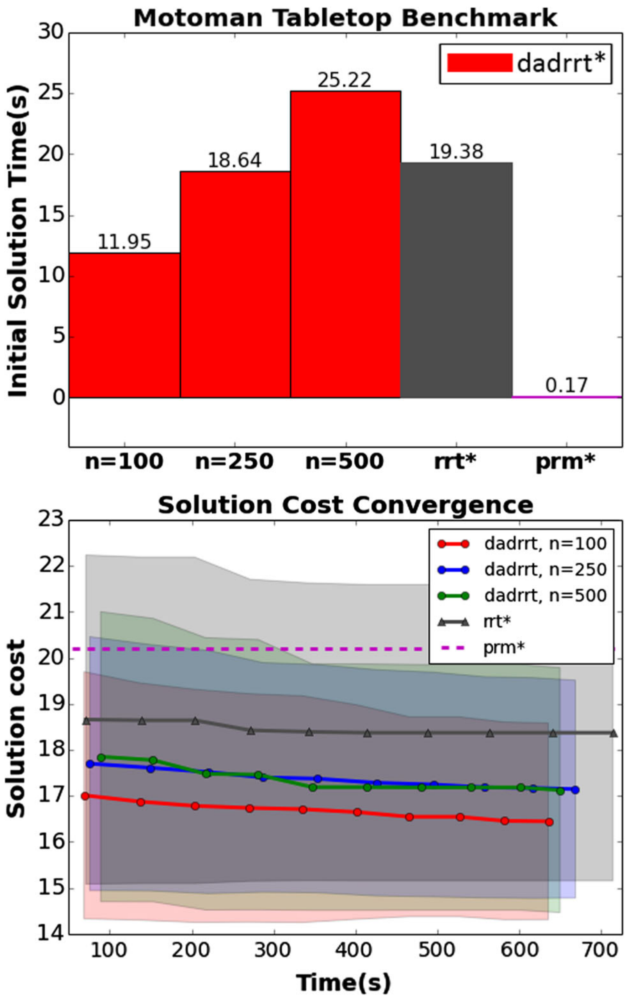


Fig. 15 Motoman Tabletop Benchmark: *Top*: The initial solution times are reported for every algorithm. *Bottom*: The average solution costs over time are reported

(*middle*). `PRM`* is still the fastest in finding solutions (only online cost considered again here). The `da-dRRT`*solution cost is much better than both the average `PRM`* solution, and `RRT`*, as shown in Fig. 16 (*bottom*). `da-dRRT`* will quickly converge for smaller roadmaps, and then stop improving the cost. The larger roadmaps contain better solutions, causing `da-dRRT`*to converge slower.

Mobile Baxter benchmark This benchmark uses a *Rethink* `Baxter` robot with a mobile base. The robot is grasping two long objects inside a shelf Fig. 17 (*top*). The robot has to navigate across a cramped, walled room, to a placing configuration inside a shelf on the other side of the room.

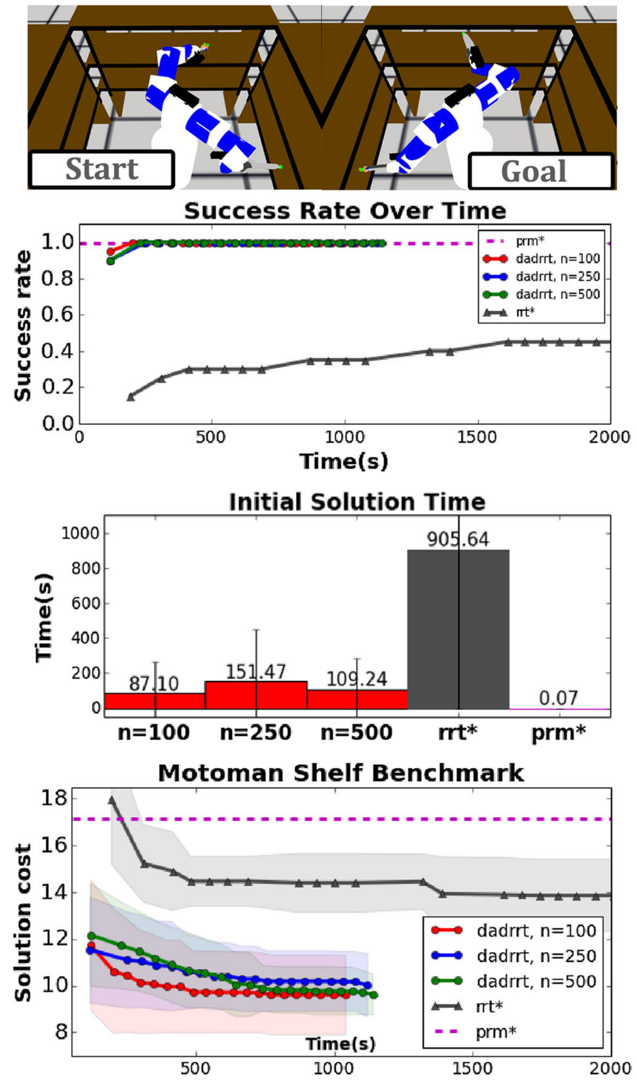


Fig. 16 Motoman Shelf Benchmark: *Top:* The setup of the benchmark. *Second:* Success ratios of the algorithms are shown over time. *Middle:* The initial solution times are reported for every algorithm. *Bottom:* The average solution costs over time are reported. The dashed horizontal line denotes the average solution cost discovered by PRM*. The shaded regions represent the corresponding algorithm's standard deviation of cost

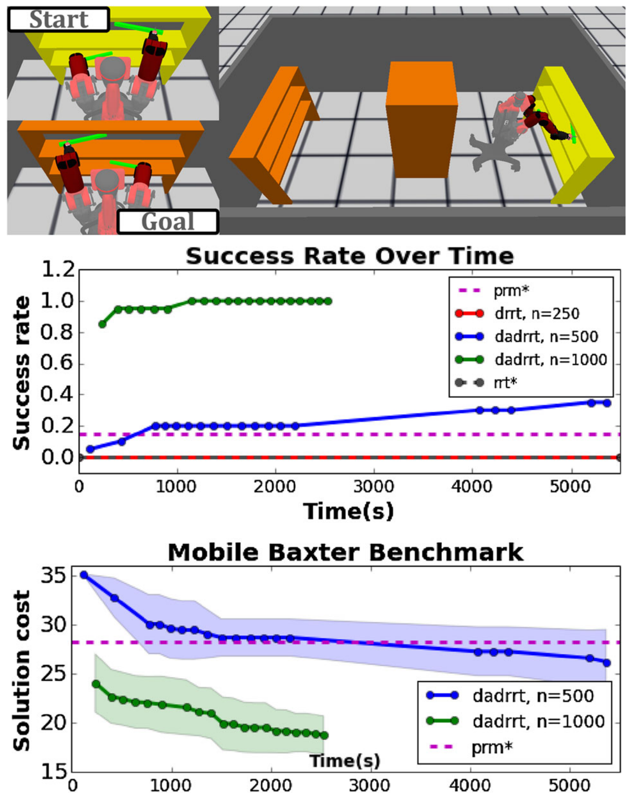


Fig. 17 Mobile Baxter Benchmark: *Top:* The setup of the benchmark. *Middle:* Success ratios of the algorithms are shown over time. *Bottom:* The average solution costs over time are reported. The dashed horizontal line denotes the average solution cost discovered by PRM*

This proves to be the most challenging problem among the three benchmarks. As shown in Fig. 17 (*middle*), RRT* fails to find a solution. It should be noted that, when tested on a simpler version of the benchmark without the pillar in the room, RRT* could find solutions. PRM* also falters by showing a very low success rate. This indicates that we need even larger roadmaps in $\mathbb{C}$ to solve harder problems. The problem is solved when a dense implicit structure, with $n = 1000$ is explored by da-dRRT*.

Figure 17 (*bottom*) shows that da-dRRT* finds better initial and converged solutions when compared to the instances in which PRM* succeeded.

8.3 Real world experiments

Experiments were performed in a 28-dimensional space with two dual-armed manipulators: (a Motoman SDA10f and a Baxter). Initial solutions were obtained in a fraction of a second for the two experimental setups, with the method allowed to run for 1000 iterations to improve the quality of the demonstrated trajectories. The two setups are chosen carefully to demonstrate in the first instance a typical application of simultaneous grasping that may arise in real world scenarios, and in the second instance a problem that forces very close interactions between the arms in close proximity.

Pre-grasp demonstration As shown in Fig. 18, the demonstration simulates an application to multi-arm manipulation, where the goals of the motion planning problem for 4 arms is to pre-grasping configurations for 4 objects placed on a table in the shared workspace between the robots. 1000 node roadmaps were constructed for each arm and dRRT* was used to search for a solution to the motion planning problem. The solution was computed offline and an open-loop execution was performed on the real system.

Coupled workspace demonstration As shown in Fig. 19, a pole is positioned between the two robots so that the arms

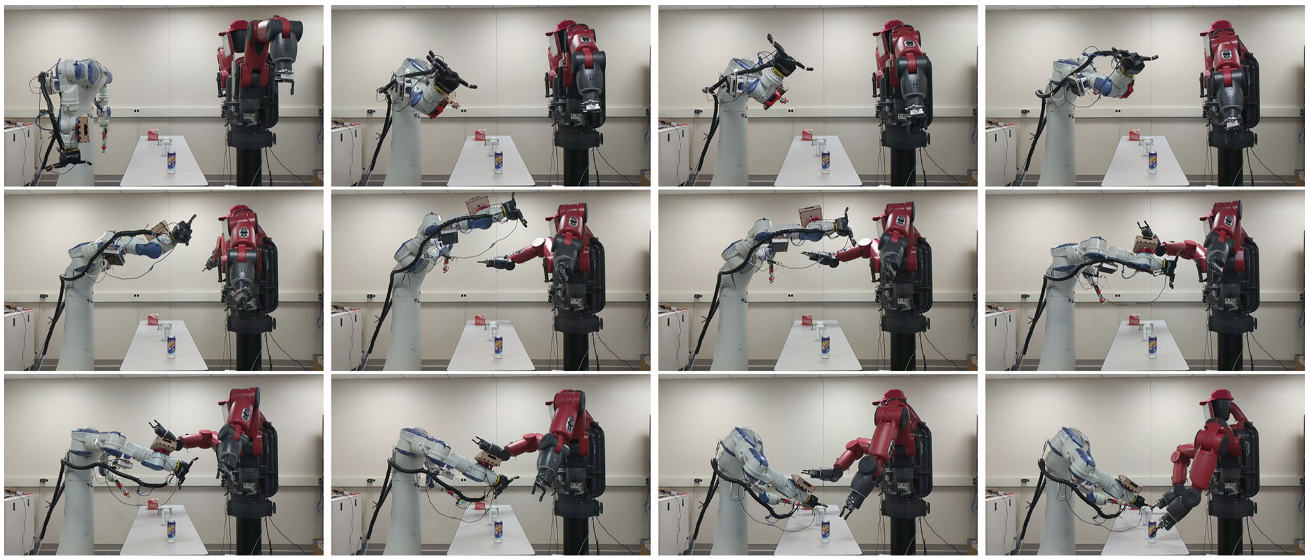


Fig. 18 Real world experiments were performed in a 28 dimensional space with 2 dual-armed manipulators planning their motion to a goal configuration corresponding to pre-grasping states for 4 objects resting on a table in the shared workspace. The sequence corresponds to freeze-frames starting in sequence from the top-left, and progressing along each row till the bottom-right

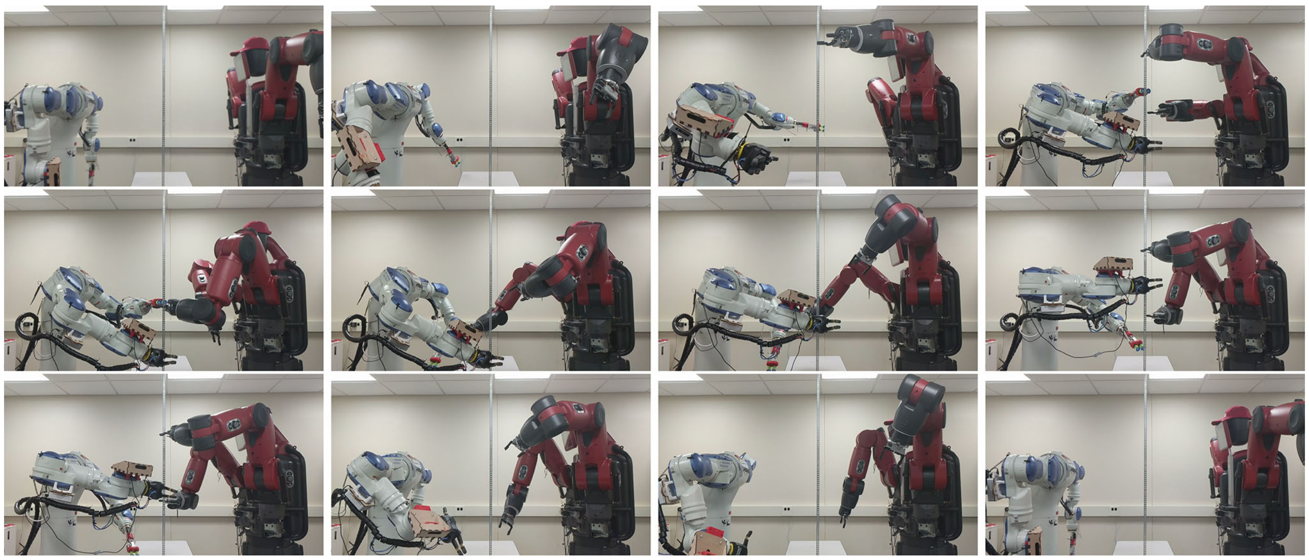


Fig. 19 Real world experiments were performed in a 28 dimensional space with 2 dual-armed manipulators planning their motion from a start state to an approach state close to a pole positioned in the center of a tightly coupled shared workspace. The arms then swap positions on the pole and return back to the start state. The goals correspond to the last images in each row. The sequence corresponds to freeze-frames starting in sequence from the top-left, and progressing along each row till the bottom-right

cannot cross over. The objective is for the 4 arms to (a) approach the pole at alternating heights, (b) then swap the height of their approaching configurations, and (c) finally return back to the start state. 1000 node roadmaps were constructed for each arm and dRRT* was used to search for a solution to the three motion planning problems. The solutions that were computed offline, were stitched together and replayed in an open loop execution on the real system.

9 Discussion

This work proves asymptotic optimality of sampling-based multi-robot planning over implicit structures by extending the dRRT approach. Asymptotic optimality is achieved by making a modification, resulting in ao-dRRT which expands a spanning tree over an implicitly defined tensor product roadmap, and leverages a simple re-wiring scheme.

This method already has the advantage of avoiding the construction of a large, dense roadmap in the composite configuration space of many robots. This can be further improved to use heuristics so as to search in an informed manner, in the dRRT* method. The method is also extended to work with robot systems, which share degrees of freedom, resulting in da-dRRT*.

Experimental results show the efficacy of the proposed approaches. Furthermore, by leveraging heuristics, dRRT* is able to solve more challenging problem instances than the baseline ao-dRRT method, and the approach is demonstrated to solve complex, real-world problems with robot manipulators operating in a shared workspace with a high degree of coupling.

In terms of practical applicability, dRRT* promises fast initial solutions times (Figs. 11, 13) on the order of a fraction of a second for most problems, including for high-dimensional, kinematically independent multi-robot problems, which is an exciting result. The solution quality improvement indicates the anytime properties of the approach, where paths of improved path quality are discovered as more computation time is invested. While problems with shared degrees of freedom provide less guidance and result in performance degradation, the scalability benefits remain even in this case relative to composite planning. Future work includes the consideration of dynamics. The existing theoretical analysis of dRRT* assumes that the individual robot systems are holonomic, which guarantees the existence of near-optimal single-robot paths (see Lemma 1 and (Janson et al. 2015, Theorem 4.1)). Recent results concerning asymptotic optimality of PRM for non-holonomic systems (see Schmerling et al. (2015a, b)) bring the hope of achieving a more general analysis of the current work as well. The proposed framework can also be leveraged toward efficiently solving simultaneous task and motion planning for many robot manipulators (Dobson and Bekris 2015). The demonstrated applications to manipulators also motivate dual-arm rearrangement challenges (Shome et al. 2018).

References

Adler, A., De Berg, M., Halperin, D., & Solovey, K. (2015). Efficient multi-robot motion planning for unlabeled discs in simple polygons. In: *IEEE transactions on automation science and engineering* (Vol. 12, pp. 1309–1317). Springer. https://doi.org/10.1109/TASE.2015.2470096, arXiv:1312.1038.

Atias, A., Solovey, K., & Halperin, D. (2017). Effective metrics for multi-robot motion-planning. In: *Proceedings of robotics: Science and systems, Cambridge, Massachusetts*. https://doi.org/10.1177/0278364918784660, arXiv:1705.10300.

Bonilla, M., Pallottino, L., & Bicchi, A. (2017). Noninteracting constrained motion planning and control for robot manipulators. In: *IEEE international conference on robotics and automation* (pp. 4038–4043). https://doi.org/10.1109/ICRA.2017.7989463.

Caccavale, F., & Uchiyama, M. (2008). Cooperative manipulators. In B. Siciliano & O. Khatib (Eds.), *Springer handbook of robotics* (pp. 701–718). Berlin: Springer. https://doi.org/10.1007/978-3-540-30301-5_30.

Canny, J. F. (1988). *The complexity of robot motion planning* (Vol. Doctoral D). Cambridge: MIT Press. https://doi.org/10.1016/j.scriptamat.2009.11.029.

Cortés, J., & Siméon, T. (2005). Sampling-based motion planning under kinematic loop-closure constraints. In: *Workshop on the algorithmic foundations of robotics* (pp. 75–90). https://doi.org/10.1007/10991541_7.

Dobson, A., & Bekris, K. E. (2013). A study on the finite-time near-optimality properties of sampling-based motion planners. In: *IEEE/RSJ international conference on intelligent robots and systems* (pp. 1236–1241). https://doi.org/10.1109/IROS.2013.6696508.

Dobson, A., & Bekris, K. E. (2015). Planning representations and algorithms for prehensile multi-arm manipulation. In: *IEEE/RSJ international conference on intelligent robots and systems* (Vol. 2015-Decem, pp. 6381–6386). https://doi.org/10.1109/IROS.2015.7354289.

Dobson, A., Solovey, K., Shome, R., Halperin, D., & Bekris, K. E. (2017) Scalable asymptotically-optimal multi-robot motion planning. In: *IEEE international symposium on multi-robot and multi-agent systems, Los Angeles, USA*.

Dogar, M., Spielberg, A., Baker, S., & Rus, D. (2015). Multi-robot grasp planning for sequential assembly operations. In: *IEEE international conference on robotics and automation* (pp. 193–200).

Erdmann, M., & Lozano-Perez, T. (1987). On multiple moving objects. *Algorithmica*, *2*(1–4), 477.

Gammell, J. D., Srinivasa, S. S., & Barfoot, T. D. (2015) BIT *: batch informed trees for optimal sampling-based planning via dynamic programming on implicit random geometric graphs. In: *IEEE international conference on robotics and automation* (pp. 3067–3074). https://doi.org/10.1109/ICRA.2015.7139620, arXiv:1405.5848v1

Gharbi, M., Cortés, J., & Siméon, T. (2009). Roadmap composition for multi-arm systems path planning. In: *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2471–2476). https://doi.org/10.1109/IROS.2009.5354415

Ghrist, R., O'Kane, J., & LaValle, S. (2005). Pareto optimal coordination on roadmaps. *Algorithmic Foundations of Robotics VI*, *24*(1), 1–16.

Gildardo, S. (2002). Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In: *IEEE international conference on robotics and automation, May* (pp. 2112–2119).

Gravot, F., & Alami, R. (2003a). A method for handling multiple roadmaps and its use for complex manipulation planning. In: *IEEE international conference on robotics and automation (Cat. No.03CH37422)* (Vol. 3, pp. 4–9). https://doi.org/10.1109/ROBOT.2003.1242038.

Gravot, F., & Alami, R. (2003b). A method for handling multiple roadmaps and its use for complex manipulation planning. In: *IEEE international conference on robotics and automation (Cat. No. 03CH37422)* (Vol. 3, pp. 4–9). https://doi.org/10.1109/ROBOT.2003.1242038.

Gravot, F., Alami, R., & Siméon, T. (2002). Playing with several roadmaps to solve manipulation problems. In: *IEEE/RSJ international conference on intelligent robots and systems* (Vol. 3, pp. 2311–2316). https://doi.org/10.1109/IRDS.2002.1041612.

Grinstead, C., & Snell, J. (2012). *Introduction to probability*. Providence, RI: American Mathmatical Society.

Harada, K., Tsuji, T., & Laumond, J. P. (2014). A manipulation motion planner for dual-arm industrial manipulators. In: *IEEE interna-*

tional conference on robotics and automation (pp. 928–934). https://doi.org/10.1109/ICRA.2014.6906965.

Hirsch, S., & Halperin, D. (2004). Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane. In: *Springer tracts in advanced robotics, Springer* (Vol. 7 STAR, pp. 239–255). https://doi.org/10.1007/978-3-540-45058-0_15.

Hopcroft, J. E., Schwartz, J. T., & Sharir, M. (1984). On the complexity of motion planning for multiple independent objects; PSPACE–hardness of the Warehousemans' problem. *International Journal of Robotics Research*, *3*(4), 76–88. https://doi.org/10.1177/027836498400300405.

Janson, L., Schmerling, E., Clark, A., & Pavone, M. (2015). Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *International Journal of Robotics Research*, *34*(7), 883–921.

Johnson, D. B. (1977). Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, *24*(1), 1–13. https://doi.org/10.1145/321992.321993.

Johnson, J. K. (2018). On the relationship between dynamics and complexity in multi-agent collision avoidance. In: *Autonomous robots, AnnArbor, Michigan* (Vol. 42, pp. 1389–1404). https://doi.org/10.1007/s10514-018-9743-4.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, *30*(7), 846–894. https://doi.org/10.1177/0278364911406761. arXiv:1105.1186.

Kavraki, L. E., Svestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, *12*(4), 566–580. https://doi.org/10.1109/70.508439.

Kloder, S., & Hutchinson, S. (2005). Path planning for permutation-invariant multi-robot formations. In: *IEEE international conference on robotics and automation* (pp. 1797–1802).

Koga, Y., & Latombe, J. C. (1994). On multi-arm manipulation planning. In: *IEEE international conference on robotics and automation* (Vol. 2, pp. 945–952). https://doi.org/10.1109/ROBOT.1994.351231.

Kornhauser, D., Miller, G., & Spirakis, P. (1984). Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In: *25th annual symposium onfoundations of computer science, 1984* (pp. 241–250). https://doi.org/10.1109/SFCS.1984.715921.

Krontiris, A., Shome, R., Dobson, A., Kimmel, A., & Bekris, K. (2015). Rearranging similar objects with a manipulator using pebble graphs. In: *IEEE-RAS international conference on humanoid robots* (Vol. 2015-Febru, pp 1081–1087). https://doi.org/10.1109/HUMANOIDS.2014.7041499.

LaValle, S., & Kuffner, J. (1999). Randomized kinodynamic planning. *IEEE International Conference on Robotics and Automation*, *20*, 378–400.

LaValle, S. M., & Hutchinson, S. A. (1998). Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, *14*(6), 912–925.

Lindemann, S. R., & LaValle, S. M. (2004). Incrementally reducing dispersion by increasing voronoi bias in rrts. In: *IEEE international conference on robotics and automation* (Vol. 4, pp. 3251–3257).

Luna, R., & Bekris, K. E. (2011). An efficient and complete approach for cooperative path-finding. In: *Twenty-fifth AAAI conference on artificial intelligence* (pp. 1804–1805).

O'Donnell, P. A., & Lozano-Pérez, T. (1989). Deadlock-free and collision-free coordination of two robot manipulators. In: *International conference on robotics and automation* (pp. 484–489). https://doi.org/10.1109/ROBOT.1989.100033.

Pelling, M. J. (1977). Formulae for the arc-length of a curve in $\mathbb{R}^N$. In R. A. Brualdi (Ed.), *The American Mathematical Monthly* (Vol. 84(6), pp. 465–467). Madison, WI: Mathematical Association of America.

Peng, J., & Akella, S. (2004). Coordinating multiple robots with kinodynamic constraints along specified paths. In J. D. Boissonat, J. Burdick, K. Goldberg, & S. Hutchinson (Eds.), *Springer tracts in advanced robotics* (Vol. 7 STAR, pp. 221–237). Berlin: Springer. https://doi.org/10.1007/978-3-540-45058-0_14.

Peng, J., & Akella, S. (2005). Multiple robots with kinodynamic constraints along specified paths. *International Journal of Robotics Research*, *24*(4), 295–310. https://doi.org/10.1177/0278364905051974.

Salzman, O., Hemmer, M., & Halperin, D. (2015). On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages. *IEEE Transactions on Automation Science and Engineering*, *12*(2), 529–538. https://doi.org/10.1109/TASE.2014.2331983. arXiv:1202.5249.

Salzman, O., Solovey, K., & Halperin, D. (2016). Motion planning for multilink robots by implicit configuration-space tiling. *IEEE Robotics and Automation Letters*, *1*(2), 760–767. https://doi.org/10.1109/LRA.2016.2524066. arXiv:1504.06631v3.

Schmerling, E., Janson, L., & Pavone, M. (2015a). Optimal sampling-based motion planning under differential constraints: The drift case with linear affine dynamics. In: *IEEE conference on decision and control* (Vol. 54rd IEEE, pp. 2574–2581). https://doi.org/10.1109/CDC.2015.7402604, arXiv:1405.7421.

Schmerling, E., Janson, L., & Pavone, M. (2015b). Optimal sampling-based motion planning under differential constraints: The drift case with linear affine dynamics. In: *IEEE conference on decision and control* (Vol. 54rd IEEE, pp. 2574–2581). https://doi.org/10.1109/CDC.2015.7402604, arXiv:1405.7421.

Shome, R., & Bekris, K. E. (2017) Improving the scalability of asymptotically optimal motion planning for humanoid dual-Arm manipulators. In: *IEEE-RAS international conference on humanoid robots* (pp. 271–277). https://doi.org/10.1109/HUMANOIDS.2017.8246885.

Shome, R., Solovey, K., Yu, J., Bekris, K., & Halperin, D. (2018). Fast, high-quality dual-arm rearrangement in synchronous, monotone tabletop setups. In: *Workshop on the algorithmic foundation of robotics*. arXiv:1810.12202v1.

Sina Mirrazavi Salehian, S., Figueroa, N., & Billard, A. (2016). Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty. In: *Robotics: Science and systems XII*. https://doi.org/10.15607/RSS.2016.XII.019.

Solovey, K., & Halperin, D. (2014). k-Color multi-robot motion planning. *International Journal of Robotic Research*, *33*(1), 82–97.

Solovey, K., & Halperin, D. (2016). On the hardness of unlabeled multi-robot motion planning. *International Journal of Robotics Research*, *35*(14), 1750–1759. https://doi.org/10.1177/0278364916672311. arXiv:1408.2260.

Solovey, K., Salzman, O., & Halperin, D. (2015a). Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *International Journal of Robotics Research*, *35*(5), 501–513. https://doi.org/10.1177/0278364915615688. arXiv:1305.2889.

Solovey, K., Yu, J., Zamir, O., & Halperin, D. (2015b). Motion planning for unlabeled discs with optimality guarantees. In: *Robotics: Science and systems*. https://doi.org/10.15607/RSS.2015.XI.011, arXiv:1504.05218.

Spirakis, P., & Yap, C. K. (1984). Strong NP-hardness of moving many disks. *Information Processing Letters*, *19*(1), 55–59.

Svestka, P., & Overmars, M. H. (1998). Coordinated path planning for multiple robots. *Robotics and Autonomous Systems*, *23*(3), 125–152.

Tang, S., & Kumar, V. (2015). A complete algorithm for generating safe trajectories for multi-robot teams. In: *International symposium on* *robotics research, sestri levante, Italy* (pp. 1–16). https://doi.org/10.1007/978-3-319-60916-4_34.

Turpin, M., Michael, N., & Kumar, V. (2013). Concurrent assignment and planning of trajectories for large teams of interchangeable robots. In: *IEEE international conference on robotics and automation* (Vol. 37, pp. 842–848). https://doi.org/10.1109/ICRA.2013.6630671.

Vahrenkamp, N., Berenson, D., Asfour, T., Kuffner, J., & Dillmann, R. (2009). Humanoid motion planning for dual-arm manipulation and re-grasping tasks. In: *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2464–2470). https://doi.org/10.1109/IROS.2009.5354625.

Vahrenkamp, N., Kuhn, E., Asfour, T., & Dillmann, R. (2010). Planning multi-robot grasping motions. In: *IEEE-RAS international conference on humanoid robots, humanoids* (pp. 593–600). https://doi.org/10.1109/ICHR.2010.5686844.

Van Den Berg, J., Guy, S. J., Lin, M., Manocha, D., Pradalier, C., Siegwart, R., et al. (2011). *Reciprocal n-Body collision avoidance, Springer tracts in advanced robotics* (star ed., Vol. 70, pp. 3–19). Berlin: Springer.

Van Den Berg, J., Snoeyink, J., Lin, M., & Manocha, D. (2009). Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In: *Robotics: Science and systems V*. https://doi.org/10.15607/RSS.2009.V.018.

Van Den Berg, J. P., & Overmars, M. H. (2005). Prioritized motion planning for multiple robots. In: *2005 IEEE/RSJ international conference on intelligent robots and systems, IROS* (pp. 2217–2222). https://doi.org/10.1109/IROS.2005.1545306.

Wagner, G., & Choset, H. (2013). Subdimensional expansion for multi-robot path planning. *Artificial Intelligence*, *219*, 1–46. https://doi.org/10.1016/j.artint.2014.11.001.

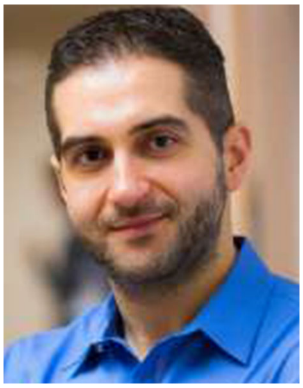
Yu, J., & LaValle, S. M. (2013). Planning optimal paths for multi-agent systems on graphs. In: *IEEE international conference on robotics and automation* (pp. 3612–3617). https://doi.org/10.1109/ICRA.2013.6631084, arXiv:1204.3830v4.

Rahul Shome is pursuing his Ph.D. in Computer Science at Rutgers University, and working with Prof. Kostas Bekris at the PRACSYS Lab. His research interests include multi-robot motion planning, manipulation planning and problems in rearrangement task planning.

Kiril Solovey is a Ph.D. student at Tel Aviv University working with Prof. Dan Halperin in the Computational Geometry Lab. His research focuses on algorithmic aspects of robotics. He is particularly interested in the design and analysis of techniques for robot motion planning. He is supported by the Clore Israel Foundation.

Andrew Dobson was born in South Lake Tahoe, California on March 26, 1988. He received his B.S. degree in Computer Science from the University of Nevada, Reno in 2010, his M.S. degree in Computer Science also from the University of Nevada, Reno in 2012, and his Ph.D. degree in Robotics from Rutgers University in 2017. Since September 2017, Andrew has been at the University of Michigan where he is a postdoc working in the Electrical and Computer Science Department. He studies underactuated control problems with Dmitry Berenson at the ARM Lab.

Dan Halperin received his Ph.D. in Computer Science from Tel Aviv University. He then spent three years at the Computer Science Robotics Laboratory at Stanford University. In 1996 he joined the School of Computer Science at Tel Aviv University, where he is currently a full professor. Halperin's main field of research is Computational Geometry and its Applications. A major focus of his work has been in research and development of robust geometric software, principally as part of the CGAL project and library. The application areas he is interested in include robotics, automated manufacturing, algorithmic motion planning and 3D printing. http://acg.cs.tau.ac.il/danhalperin.



Kostas E. Bekris is an Associate Professor in the Computer Science department of Rutgers, the State University of New Jersey. He received his MS and PhD degrees in Computer Science from Rice University in 2004 and 2008 respectively. He was an Assistant Professor at the Department of Computer Science and Engineering at the University of Nevada, Reno from 2008 to 2012. He is the recipient of a NASA Early Career Faculty award and his research has been supported by NSF, NASA, DHS and the DoD. His research interests include planning and coordination of robots, especially for systems with many degrees of freedom and significant dynamics, as well as applications to robotic manipulation, planetary exploration, cyber-physical systems and physically-realistic virtual agents.