Improving the Scalability of Asymptotically Optimal Motion Planning for Humanoid Dual-arm Manipulators

Rahul Shome Kostas E. Bekris

Abstract— **Due to high-dimensionality, many motion planners for dual-arm systems follow a decoupled approach, which does not provide guarantees. Asymptotically optimal sampling-based planners provide guarantees but in practice face scalability challenges. This work improves the computational scalability of the latter methods in this domain. It builds on top of recent advances in multi-robot motion planning, which provide guarantees without having to explicitly construct a roadmap in the composite space of all robots. The proposed framework builds roadmaps for components of a humanoid robot's kinematic chain. Then, the tensor product of these component roadmaps is searched implicitly online in a way that asymptotic optimality is provided. Appropriate heuristics from the component roadmaps are utilized for discovering the solution in the composite space effectively. Evaluation on various dual-arm problems show that the method returns paths of increasing quality, has significantly reduced space requirements and improved convergence rate relative to the standard asymptotically optimal approaches.**

I. Introduction

The dual-arm form of humanoid robots brings the promise of efficient operation in spaces designed for people, both in industrial and domestic setups, which can often be complex and cluttered environments [34]. For this reason, motion planners are needed that deal with dual-arm humanoid systems and achieve high-quality paths in a computationally efficient manner. The focus here is on tasks where the two arms move independently and not when they transfer the same object or when they are linked via a virtual constraint.

Planning for such systems is challenging due to high dimensionality and the coupled nature of the involved degrees of freedom (DoF). For instance, the robot on the left of Fig. 1 has a rotational DoF at the torso, which affects the placement of both arms. A robot such as Baxter on a mobile base (right side of Fig. 1) will have even more shared DoFs arising from the mobile base.

As motion planning is computationally hard, it is difficult to achieve both computational efficiency and high-quality solutions for humanoid robots. The problem is more involved when the torso or the mobile base need to be coordinated with the arms. This requires operating in the full $\mathbb{C}$-space of the humanoid robot and complicates the application of decoupled solutions [29], [33], which do not provide guarantees. Applying sampling-based planners in the full $\mathbb{C}$-space can provide guarantees, such as asymptotic optimality [19], [20], [25], but their efficiency and convergence rate are negatively impacted by high-dimensionality. Most importantly, the space requirements of constructing asymptotically optimal roadmaps in the high-dimensional $\mathbb{C}$-spaces of humanoid robots renders them often impractical.

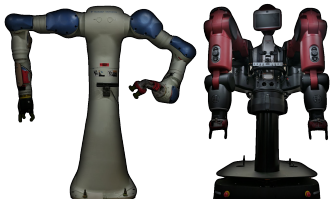

Fig. 1. Example dual-arm robots with shared DoF: (left) Yaskawa Motoman SDA10F with a torsional DoF and (right) a Baxter robot on a mobile base. Roadmaps can be computed for a decomposition of such robots' kinematic chain, which are explored quickly online to provide paths of increasing quality.

This work aims to provide an asymptotically optimal motion planner for dual-arm systems that is more computationally efficient than standard sampling-based planning in the full $\mathbb{C}$-space. The motivation for the approach arises from recent insights in multi-robot motion planning [35], where it has been shown that asymptotic optimality is possible without having to explicitly construct a roadmap in the composite space of all robots [10]. In particular, given asymptotically optimal sampling-based roadmaps for each individual robot, it is possible to search in an implicit manner the tensor product of all these roadmaps [36], without explicitly constructing it, and still achieve asymptotic optimality. This has major scalability benefits as it significantly reduces space requirements, results in faster solution times and better convergence rate.

The key premise of the current paper is to apply this implicit search idea over a tensor product roadmap for dual-arm manipulators. A challenge, however, in this case is that the DoFs of the different arms in a humanoid robot are coupled through the torsional or mobile base DoFs. The solution is to utilize a natural decomposition of the kinematic chain of the dual-arm system, which considers the shared DoFs [12], and build asymptotically optimal roadmaps for the individual components. Then, the method implicitly searches online the resulting tensor product roadmap in a way that asymptotic optimality is achieved. There are some increased requirements for online collision checking relative to the case of multiple independent robots [10] but effective heuristics, which can be computed easily on the component roadmaps help to guide the search process in the most promising part of the full $\mathbb{C}$-space. The current paper also generalizes the theoretical results of the prior effort so that the path quality guarantees can be provided for a Euclidean metric, which is more appropriate for a humanoid robot than the sum of distances of individual robots considered in prior work [10].

R. Shome and K. E. Bekris are with the Computer Science Dept. of Rutgers University, NJ, USA, {rahul.shome, kostas.bekris}@cs.rutgers.edu.

This work is supported by NSF awards CCF-1330789, IIS-1451737 and IIS-1617744. Any opinions or findings expressed in this paper do not necessarily reflect the views of the sponsor.

The experimental evaluation highlights the computational benefits of the approach for dual-arm motion planning challenges of increasing difficulty. The method is compared against asymptotically optimal sampling-based planners that search directly the full $\mathbb{C}$-space of the system, i.e., both PRM* and RRT*. It is shown that the proposed decomposition and online exploration of the resulting tensor roadmap indeed result in significantly smaller space requirements, faster convergence to high-quality solutions and overall improved computational efficiency.

II. Related Work

Early results established the hardness of motion planning [3]. These methods can be applied to dual-arm systems:

- Potential fields [21] can be applied to control dual-arm systems [37] but without formal guarantees.
- This is a similar limitation of trajectory optimization [41], which has been applied to humanoid locomotion [24].
- Heuristic search can be used but guarantees are up to the discretization resolution [4], [5].

Sampling-based motion planners, such as the PRM [20], construct a roadmap of collision-free configurations through sampling and then search over it. The early methods could achieve only probabilistic completeness but as long as the connectivity of the underlying roadmap is sufficient, then asymptotic optimality can also be achieved with methods like PRM* and RRT* [19]. Nevertheless, high dimensional $\mathbb{C}$-spaces pose a significant challenge and result in slower convergence.

Searching over the tensor product of multiple roadmaps, each designed for an individual robot, was an idea that was explored early [36], including for multi-arm motion planning [12] but without guarantees for path quality. Meanwhile, progress was made in efficient search strategies over tensor roadmaps [40], which motivated a strategy for implicitly exploring the tensor roadmap by building a tree-like structure over it, similar to an RRT [25], in a way that preserved probabilistic completeness [35]. This can be adapted to an RRT* like search [19] and provide asymptotic optimality [10]. These strategies do not effectively deal with shared DoFs, which are common in many dual-armed humanoid platforms.

One way to achieve improved performance in multi-arm planning is to follow a decoupled methodology inspired by multi-robot strategies [26] instead of searching the full $\mathbb{C}$-space. This involves computing paths for each arm independently and then coordinating them through "velocity tuning" [22], [29], [33] or some form of prioritization [42]. These methods cede completeness and optimality guarantees. Furthermore, the decoupling is not straightforward when shared DoFs between the two arms must also be controlled.

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 [13], [14]. The topology of dual-arm manipulation has been formalized [16], [23] and extended to the N-arm case [8]. It requires the consideration of multi-robot grasp planning [11], [39], regrasping [38], as well as closed kinematic chain constraints [1], [7]. Furthermore, force control strategies are helpful for multi-arm manipulation of a common object [2]. Recently coordinated control has been applied to solve human-robot interaction tasks [28].

III. Problem Setup and Notation

Consider a dual-arm manipulator in a workspace $\mathbb{W}$. The robot's configuration space $\mathbb{C}_{full} \subset \mathbb{R}^d$ is defined as the Cartesian product of the robot's d degrees of freedom (DoFs). A point q in that space will be a configuration $q \in \mathbb{C}_{full}$. The set of obstacles $\mathbb{O}$ in $\mathbb{W}$ map to the set of colliding configurations for the robot $\mathbb{C}_{obs} \subset \mathbb{C}_{full}$. Then, $\mathbb{C}_{free}$ corresponds to: $\mathbb{C}_{free} = \mathbb{C}_{full} - \mathbb{C}_{obs}$.

As shown in Fig. 2, the robot's DoFs can be grouped intro a left, right and a shared DoF subset, so that: $\mathbb{C}_{full} = \mathbb{C}_l \times \mathbb{C}_s \times \mathbb{C}_r$. Then, a projection operator $q \rightarrowtail \mathbb{C}_{subset}$ maps points $q \in \mathbb{C}$ to points $q' \in \mathbb{C}_{subset}$, where $\mathbb{C}_{subset}$ is a lower-dimensional subset of $\mathbb{C}$ and the point q' has the same values as q for the DoFs of $\mathbb{C}_{subset}$. With slight abuse of notation the projection operator will also be used over a set of points in $\mathbb{C}$.

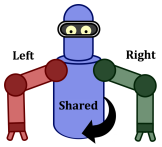

Fig. 2. $\mathbb{C}_{full} = \mathbb{C}_l \times \mathbb{C}_s \times \mathbb{C}_r$.

Given configurations $q_{start} \in \mathbb{C}_{free}$ and $q_{goal} \in \mathbb{C}_{free}$, a candidate path $\Sigma : [0,1] \rightarrow \mathbb{C}_{free}$ is a continuous curve in $\mathbb{C}_{free}$, such that $\Sigma(0) = q_{start}$ and $\Sigma(1) = q_{goal}$. A candidate path $\Sigma(i)$ can be decomposed to projections $[\Sigma_l(i), \Sigma_s(i), \Sigma_r(i)]$ along $\mathbb{C}_l$, $\mathbb{C}_s$ and $\mathbb{C}_r$ respectively.

The objective is to find a path $\Sigma^* = \arg\min_\Sigma c(\Sigma)$ which minimizes a *cost function* defined to be the standard arc-length of the path given a *Euclidean* metric in $\mathbb{C}_{full} \subset \mathbb{R}^d$:

$$c(|\Sigma|) = \|\Sigma\| = \int \sqrt{(f_1'(t)^2 + \cdots + f_d'(t)^2)}\, dt, \qquad (1)$$

where $f_i(t)$ is the coordinate function of the curve Σ along the i-th DoF, where $t \in [0,1]$ and $i \in [1,d]$. The coordinate function is assumed to be continuous with respect to the Lebesque measure on t and of bounded variation.

IV. Solution Framework

Given the hardness of the problem, the objective of discovering a Σ^* path is relaxed to having an approach that eventually converges to such a path, i.e., achieves *asymptotic optimality* (AO). This can be achieved by extending a tree data structure $\mathbb{T}$ $(\mathbb{V}, \mathbb{E})$ (as in RRT*) or a graph/roadmap $\mathbb{G}(\mathbb{V}, \mathbb{E})$ in the entire $\mathbb{C}_{free}$ (as in PRM*) through a sampling-based process [19]. Such algorithms sample collision-free configurations $v \in \mathbb{C}_{free}$ and store them as nodes $\mathbb{V}$ in the corresponding data structure. Then, edges between two neighboring nodes are defined as the local shortest path, as long as that path lies in $\mathbb{C}_{free}$. For the methods to be AO, a sufficient neighborhood radius needs to be considered.

Theorem 1 (Asym. Optimality of RRT*/ PRM* [19]): For the PRM* (or RRT*) algorithm: $\mathbb{P}(\liminf_{n\to\infty} A_n) = 1$ and $\mathbb{P}(\limsup_{n\to\infty} A_n^c) = 0$, where $\mathbb{P}$ is the probability of event A_n that the roadmap $\mathbb{G}$ (or $\mathbb{T}$) contains an optimal path Σ^* at iteration n. A_n^c is the complementary event.

Practically, however, the performance of these algorithms suffers for high-dimensional systems, such as dual-arm

robots. Moreover, it becomes difficult to even store a large enough roadmap that is built with large enough neighborhood radius. RRT* does not suffer from the same memory issue as it keeps only the best edge for each node but has increased online cost as the collision checking takes place during query resolution. This work builds PRM* roadmaps in lower dimensional projections of the dual-arm system - so as to take advantage of preprocessing, while keeping memory requirements low - and implicitly explores the tensor product of these roadmaps in a way that provides asymptotic optimality. In particular, as shown in Fig. 3, the proposed framework constructs:

- 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 do not refer to the shared DoFs. So, only self-collisions between arm links are avoided.

A tensor product $\hat{\mathbb{G}}_{ij}(\hat{\mathbb{V}}_{ij}, \hat{\mathbb{E}}_{ij})$ is defined over two roadmaps $\mathbb{G}_i(\mathbb{V}_i, \mathbb{E}_i)$ and $\mathbb{G}_j(\mathbb{V}_j, \mathbb{E}_j)$, such that $\hat{\mathbb{V}}_{ij} = \mathbb{V}_i \times \mathbb{V}_j$ and $[(v_i \times v_j), (v_i' \times v_j')] \in \hat{\mathbb{E}}_{ij}$ as long as $((v_i, v_i') \in \mathbb{E}_i \vee v_i = v_i')$ and $((v_j, v_j') \in \mathbb{E}_j \vee v_j = v_j')$. This definition allows for self-edges in the tensor roadmap where some DoFs remain static. The method focuses on the tensor product roadmaps: $\hat{\mathbb{G}}_l = \mathbb{R}_{ls} \times \mathbb{P}_r$, and $\hat{\mathbb{G}}_r = \mathbb{R}_{sr} \times \mathbb{P}_l$. Searching $\hat{\mathbb{G}}_l$ or $\hat{\mathbb{G}}_r$ corresponds to searching over $\mathbb{G}$ in $\mathbb{C}_{free}$. The proposed method does not explicitly construct these tensor products, but implicitly searches them only given access to the lower-dimensional component roadmaps.

V. Method

This section describes the proposed method, summarized in Algorithm 1, which builds a tree $\mathbb{T}$ over both $\hat{\mathbb{G}}_l$ and $\hat{\mathbb{G}}_r$. It is sufficient to consider only one roadmap, but in practice, interaction between the two parts of the tree helps the convergence, since we can evaluate additional solutions and rewires. Implicitly searching tensor product roadmaps makes it possible to guide the search given information from the constituent roadmaps. The approach shows improved performance relative to an RRT* constructed in $\mathbb{C}_{full}$.

Algorithm 1: da_dRRT*$(\hat{\mathbb{G}}_l, \hat{\mathbb{G}}_r, \mathtt{S}, \mathtt{T})$

1 $\hat{\mathbb{G}}_l$.Add_Verts$(\{S_l, T_l\})$; $\hat{\mathbb{G}}_r$.Add_Verts$(\{S_r, T_r\})$;
2 $S_l \leftarrow \mathbb{T}$.Init$(S, \hat{\mathbb{G}}_l)$; $S_r \leftarrow \mathbb{T}$.Init$(S, \hat{\mathbb{G}}_r)$;
3 $\mathbb{T}$.Add_Vertex(S_l); $\mathbb{T}$.Add_Vertex(S_r);
4 $\pi_{\text{best}} \leftarrow \emptyset$; $v \leftarrow \emptyset$; $iterations = 0$;
5 **while** $iterations < max\_iters$ **do**
6 $\quad v \leftarrow$ Expand_da_dRRT*$(\hat{\mathbb{G}}_l, \hat{\mathbb{G}}_r, \mathbb{T}, v, T)$;
7 $\quad \pi \leftarrow$ Trace_Path$(\mathbb{T}, S, T)$;
8 $\quad$ **if** $\pi \neq \emptyset \cap cost(\pi) < cost(\pi_{\text{best}})$ **then** $\pi_{\text{best}} \leftarrow \pi$;
9 $\quad iterations$++;
10 **return** π_{best}

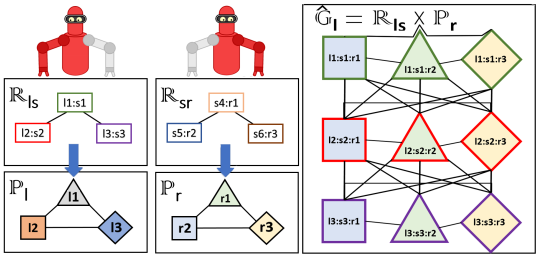

Fig. 3. The image on the left is an illustration of the decomposition of the space to create arm-shared DoF roadmaps $\mathbb{R}$ and arm only roadmaps $\mathbb{P}$. The example has three vertices in each roadmap consisting of a combination Left(l), Shared(s), and Right(r) values. Similar to the accompanying implementation, the vertices on the arm-only roadmaps are derived from the vertices in the corresponding $\mathbb{R}$ 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 $\hat{\mathbb{G}}_r$.

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 single tree 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 arose from. 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 the Expand_da_dRRT* function. The solution path is updated (Line 8) if a better cost path is discovered during the iteration.

Expansion Step: The expansion step (Algorithm 2) selects a vertex on the tree x^{near} to expand. An oracle function $\mathbb{O}_d$ is used to select a vertex x^{n} from the neighbors of x^{near} on the tensor roadmap. The algorithm attempts to add the collision free trajectory to x^{n} with the least cost from the start. It then rewires the neighborhood of x^{n} .

Algorithm 2: Expand_da_dRRT*$(\hat{\mathbb{G}}_l, \hat{\mathbb{G}}_r, \mathbb{T}, x^{\text{last}}, T)$

1 **if** $x^{\text{last}} == \emptyset$ **then**
2 $\quad x^{\text{rand}} \leftarrow$ Random_Sample();
3 $\quad x^{\text{near}} \leftarrow$ Nearest_Neighbor$(\mathbb{T}, x^{\text{rand}})$;
4 $\quad x^{\text{n}} \leftarrow \mathbb{O}_{\text{d}}(x^{\text{near}}, x^{\text{rand}}, \hat{\mathbb{G}}_l, \hat{\mathbb{G}}_r, T)$;
5 **else**
6 $\quad x^{\text{n}} \leftarrow \mathbb{O}_{\text{d}}(x^{\text{last}}, T, \hat{\mathbb{G}}_l, \hat{\mathbb{G}}_r, T)$;
7 $N \leftarrow$ Neighborhood$(x^{\text{n}}, \mathbb{T})$;
8 $x^{\text{best}} \leftarrow \arg\min_{v \in N \mid \mathbb{L}(.) \subset \mathbb{C}_{free}} c(v) + c(\mathbb{L}(v, x^{\text{n}}))$;
9 **if** $x^{\text{best}} == \emptyset$ **then return** $\emptyset$;
10 **if** $x^{\text{n}} \notin \mathbb{T}$ **then**
11 $\quad \mathbb{T}$.Add_Vertex(x^{n}); $\mathbb{T}$.Add_Edge$(x^{\text{best}}, x^{\text{n}})$;
12 **else** $\mathbb{T}$.Rewire$(x^{\text{best}}, x^{\text{n}})$;
13 **for** $v \in N$ **do**
14 $\quad$ **if** $c(x^{\text{n}}) + c(\mathbb{L}(x^{\text{n}}, v)) < c(v) \cap \mathbb{L}(x^{\text{n}}, v) \subset \mathbb{C}_{free}$ **then** $\mathbb{T}$.Rewire(x^{n}, v) ;
15 **if** Heuristic$(x^{\text{n}}) <$ Heuristic(x^{best}) **then**
16 $\quad$ **return** x^{n} ;
17 **else return** $\emptyset$;

The structure of the expansion step is similar to RRT*. The major difference is the use of the Oracle function $\mathbb{O}_d$ to determine the expansion of the tree to x^{n} . The $\mathbb{O}_d$

restricts the generation of x^{n} in such a way that $x^{\text{n}} \in \hat{\mathbb{G}}_l.\hat{\mathbb{V}}$, or $x^{\text{n}} \in \hat{\mathbb{G}}_r.\hat{\mathbb{V}}$. This ensures that the tree exploration is implicitly guided by the tensor product roadmap connectivity.

The proposed method can take advantage of the preprocessing that has taken place during the generation of the constituent roadmaps. Certain collision checks can be avoided during the tree expansions. When nodes are adjacent over a roadmap $\mathbb{R}$, then collision checks of the arm corresponding to $\mathbb{R}$ and the static obstacles can be ignored. Similarly, for both $\mathbb{R}$ and $\mathbb{P}$, self collisions can be ignored for the arms.

The function `Expand_da_dRRT*` also employs a process for balancing exploration and exploitation. When x^{last} is $\emptyset$, the algorithm performs exploration similar to `RRT*`. The method selects the nearest neighbor on the tree x^{near} to a random sample x^{rand} (Lines 2, 3). The node to add x^{n} is generated by the Oracle function $\mathbb{O}_d$ (Alg. 3 - Lines 4, 6). The method then proceeds to generate the neighborhood N of x^{n} (Line 7). A parent x^{best} is selected from N based on the shortest collision free trajectory to reach x^{n} through the neighborhood N (Line 8). If such a collision-free way to add x^{n} exists, it is added to the tree or rewired (Lines 10-12). An additional rewiring step is attempted if there exists a shorter collision free path to reach a neighbor in N, through x^{n} (Lines 13-14). The algorithm uses goal-biasing whenever x^{n} has a better heuristic measure than its parent. During exploitation, heuristics are used in $\mathbb{O}_d$ to guide the generation of the roadmap neighbors. In the current implementation, the `Cost_To_Goal` heuristic is obtained for $\mathbb{R}$ by using the *Johnson's* algorithm [18] to precompute pairwise shortest path costs over the graph. The neighborhood N for x^{n} considers the tensor roadmap neighborhoods that are part of the tree for both roadmaps. Then, $x^{\hat{\text{n}}}$ is chosen to be the nearest tree vertex that was generated from the other roadmap. N is the set of all tree vertices that are tensor roadmap neighbors of x^{n} or $x^{\hat{\text{n}}}$.

Oracle Function: The oracle function $\mathbb{O}_d$ (Algorithm 3), takes as input the tree vertices x^{near} and x^{rand} . If x^{rand} is not the target, then a random neighbor is selected over the tensor roadmap as $[r^{new}, p^{new}]$. `Combine` generates the composite state in $\mathbb{C}_{full}$ from the roadmap and projection nodes.

Computation of Arm Roadmaps: As an implementation choice, the nodes of the arm-only roadmaps $\mathbb{P}_l$ and $\mathbb{P}_r$ are projections of the nodes of the arm-shared roadmaps, i.e., $\mathbb{P}_l.\mathbb{V} = \{\mathbb{R}_{ls}.\mathbb{V} \rightarrowtail \mathbb{C}_l\}$, and $\mathbb{P}_r.\mathbb{V} = \{\mathbb{R}_{sr}.\mathbb{V} \rightarrowtail \mathbb{C}_r\}$. The edges account for the projected space's connection radius considering only arm self-collisions.

VI. Analysis

The analysis of this section extends the analysis of the `dRRT*` algorithm, which has been developed in the context of multi-robot motion planning [10]. In particular, it is shown that for the *Euclidean* metric the implicit tensor product roadmaps $\hat{\mathbb{G}}_l$ and $\hat{\mathbb{G}}_r$ converge asymptotically to a robustly optimal solution in $\mathbb{C}_{free}$. This is different from the analysis of `dRRT*`, which has been developed for a multi-robot metric. The *Euclidean* metric is more appropriate for dual-arm motion planning, due to the coupled nature of the problem. Furthermore, the `da_dRRT*` algorithm is shown to converge to the optimal solution that exists on the tensor product structure. This is shown for the oracle function with random neighborhood selection and an informed roadmap heuristic. The following sections provide an analysis over the tensor product roadmap $\hat{\mathbb{G}}_l = \mathbb{R}_{ls} \times \mathbb{P}_r$. The same results hold for $\hat{\mathbb{G}}_r$. All instances of the connection radius $r(n)$ are assumed to correspond to the $\mathbb{C}$-space that it is applied to, so that AO guarantees are achieved in the corresponding space.

Algorithm 3: $\mathbb{O}_{\text{d}}(x^{\text{near}}, x^{\text{rand}}, \hat{\mathbb{G}}_l, \hat{\mathbb{G}}_r, T)$

1 **if** $x^{\text{near}} \in \hat{\mathbb{G}}_l$ **then** $\mathbb{G} \leftarrow \hat{\mathbb{G}}_l$;
2 **if** $x^{\text{near}} \in \hat{\mathbb{G}}_r$ **then** $\mathbb{G} \leftarrow \hat{\mathbb{G}}_r$;
3 $\mathbb{R} \leftarrow \mathbb{G}.\mathbb{R}$; $\mathbb{P} \leftarrow \mathbb{G}.\mathbb{P}$;
4 $r \leftarrow x^{\text{near}} \rightarrowtail \mathbb{R}.\mathbb{C}$; $p \leftarrow x^{\text{near}} \rightarrowtail \mathbb{P}.\mathbb{C}$;
5 **if** $x^{\text{rand}} != T$ **then**
6 $\quad r^{new} \leftarrow$ `Random_Neighbor`$(\mathbb{R}, r)$;
7 $\quad p^{new} \leftarrow$ `Random_Neighbor`$(\mathbb{P}, p)$;
8 **else**
9 $\quad T_r \leftarrow T \rightarrowtail \mathbb{R}.\mathbb{C}$; $T_p \leftarrow T \rightarrowtail \mathbb{P}.\mathbb{C}$;
10 $\quad r^{new} \leftarrow \arg\min_{x \in \texttt{Adj}(r,\mathbb{R})}$ `Cost_To_Goal`$(x, T_r, \mathbb{R})$;
11 $\quad p^{new} \leftarrow \arg\min_{x \in \texttt{Adj}(p,\mathbb{P})}$ `Heuristic`$(x, T_p, \mathbb{P})$;
12 **return** `Combine`$(\mathtt{r}^{\mathtt{new}}, \mathtt{p}^{\mathtt{new}})$

A path $\Sigma : [0,1] \rightarrow \mathbb{C}_{free}$ is δ-robust if the distance from any point on the path to a colliding configuration is at least δ, i.e., $||\Sigma(\tau) - c_{obs}|| > \delta,\ \forall\ \tau \in [0,1],\ \forall\ c_{obs} \in \mathbb{C}_{obs}, \delta > 0$.

A path Σ^* is robustly optimum if $||\Sigma^*|| = c^*,\ c^* = \texttt{infimum}\{c : c = ||\Sigma||, ||\Sigma^*|| < (1+\epsilon)c\}$. Then, a roadmap is asymptotically optimal if $||\Sigma^{(n)}|| \leq (1+\epsilon)||\Sigma^*||, \forall \epsilon > 0$, as $n \rightarrow \infty$, where $\Sigma^{(n)}$ is the solution path in the roadmap at iteration n.

Asymptotic optimality of tensor roadmaps: Given the assumption that $\mathbb{R}_{ls}$ and $\mathbb{P}_r$ are `AO`, the robust optimality of the tensor roadmap can be argued [32]. For $\mathbb{R}_{ls}$ and $\mathbb{P}_l$: $r(n) \geq r^*(n) = 2(1+\eta)(\frac{1}{d})^{\frac{1}{d}}(\frac{logn}{n})^{\frac{1}{d}}$ [17].

It is shown that the robustness of Σ in $\hat{\mathbb{G}}_l$, implies the robustness of the constituent (Σ_{ls}, Σ_r) in $\mathbb{C}_{ls}$ and $\mathbb{C}_r$. Consider any $X = (\Sigma_{ls}(\tau), x_r)$, where x_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, X is a colliding configuration: $\delta \leq \|\Sigma(\tau) - X\|$. But X and $\Sigma(\tau)$ only differ in x_r, so $\delta \leq ||\Sigma_r(\tau) - x_r||$, i.e., the path Σ_r also has clearance δ.

By switching the decomposition of Σ in $\hat{\mathbb{G}}_l$ into (Σ_l, Σ_{sr}), by the above reasoning: $\delta \leq ||\Sigma(\tau) - X|| \implies \delta \leq ||\Sigma_l(\tau) - x_l||$. Now, since $\forall \tau$: $||\Sigma_{ls}(\tau) - x_{ls}|| \geq ||\Sigma_l(\tau) - x_l||$, $\delta \leq ||\Sigma_{ls}(\tau) - x_{ls}||$ holds, proving robustness for Σ_{ls}.

Given the decomposition, $\mathbb{C}_{full}$ is divided into two parts: $\mathbb{R}_{ls}$ and $\mathbb{P}_r$. Without loss of generality, rename them as $\mathbb{G}_0$ and $\mathbb{G}_1$. If a robust cost optimal path Σ^* exists in $\mathbb{C}_{full}$, given the properties of $\mathbb{G}_i$, they individually contain $\Sigma_i^{(n)}$ after n iterations, converging to Σ_i^* as $n \rightarrow \infty$:

$$|\Sigma_i^{(n)}| \leq (1 + o(1))|\Sigma_i^*| \tag{2}$$

It is shown that since the constituent solution paths $\Sigma_i^{(n)}$ converge, then their composition that exists in $\mathbb{C}_{full} : \Sigma^{(n)} =$

$(\Sigma_1^{(n)}, \ldots, \Sigma_R^{(n)})$, also converges to the optimal cost $\|\Sigma^*\|$.

Convergence for the Euclidean Metric : As defined in Eq. 1, the arc length of a path is defined as $\|\Sigma\| = s = \int (f_1'(t)^2 + \cdots + f_d'(t)^2)^{\frac{1}{2}} dt$. Since, for solution paths, $\Sigma(1)$ exists, Σ is a *rectifiable* curve. By definition and assuming smoothness and bounded variation [30],

$$s(x) = \sup_P \sum_{j=1}^{m} \left(\sum_{i=1}^{d} (f_i(t_j) - f_i(t_{j-1}))^2 \right)^{\frac{1}{2}} \quad (3)$$

over all partitions $P : 0 = t_0 < \cdots < t_m = x$, m is finite.

Let P^* be the *finite* supremum partitioning over t, that has m parts with finite measure. A finer partitioning $Q \supseteq P^*$ (Q contains all the partitions of P^* but also additional ones), would keep the arc length the same. Given every part has a positive measure of $s(P(j)) - s(P(j-1))$, $\exists\ \theta^* = \sup\{\theta | \frac{s(P^*(j)) - s(P^*(j-1))}{\theta} \in N_+, \forall j\}$. A partition with every part of equal measure θ^* is a finer partitioning compared to P^*. Define $\omega = \frac{s(1)}{\theta^*}$, such that ω_i for each Σ_i, $\omega_i^{(n)}$ for each $\Sigma_i^{(n)}$, Ω_i for each Σ, and $\Omega_i^{(n)}$ for each $\Sigma^{(n)}$. These values provide the number of parts in every trajectory partition. Find now a finer partition that allows the same number of parts in each trajectory. It can be shown that $\exists\ L = \inf\{L' | \frac{L'}{\omega} \in N_+, \forall \omega \in \{\omega_i, \omega_i^{(n)}, \Omega_i, \Omega_i^{(n)}\}\}$.

Claim 1: This selects the smallest *finite* L, such that all the trajectories can be further partitioned into the *equal number* of *equal measure* parts, while preserving Eq. 3.

Discretizing the trajectories $\Sigma_i^{(n)}$, and Σ_i^*, into L pieces of equal length, yields two trajectory sequences, for $l \in N_+, l \leq L$. The measure of every partition is denoted by $\Sigma_i(l)$ such that $\|\Sigma_i(l)\| = \frac{\|\Sigma_i\|}{L}$. Using the relation from Eq. 2 for the trajectory sequences, for any finite L, it can be shown that the trajectory in the tensor product roadmap achieves the same convergence as the composite roadmaps.

$$\|\Sigma_i^{(n)}\| \leq (1 + o(1))\|\Sigma_i^*\|$$

$$\Rightarrow \sum_{l=1}^{L} \|\Sigma_i^{(n)}(l)\| \leq (1 + o(1)) \sum_{l=1}^{L} \|\Sigma_i^*(l)\|$$

$$\Rightarrow \|\Sigma_i^{(n)}(l)\|^2 \leq (1 + o(1))^2 \|\Sigma_i^*(l)\|^2$$

$$\Rightarrow \sum_{i=1}^{R} \|\Sigma_i^{(n)}(l)\|^2 \leq (1 + o(1))^2 \sum_{i=1}^{R} \|\Sigma_i^*(l)\|^2$$

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

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

$$\Rightarrow \|\Sigma^{(n)}\| \leq (1 + o(1))\|\Sigma^*\| \quad \text{(Using Eq.3 and Claim 1)}$$

Convergence of `da_dRRT*`: Given the implicit roadmaps $\hat{\mathbb{G}}_l$, and $\hat{\mathbb{G}}_r$, composed of n samples each, the algorithm shall discover a solution $\Sigma^{(n,m)}$, after m iterations. It can be shown that $\lim_{n,m\to\infty} \Pr[\|\Sigma^{(n,m)}\| \leq (1+\epsilon)c^*] = 1,\ \forall \epsilon > 0$.

During every iteration of the algorithm, there is a volume $Vol(x^{\text{near}})$ around tree vertex x^{near}, such that $\Pr[select(x^{\text{near}})] \propto \frac{\mu(Vol(x^{\text{near}}))}{\mu(\mathbb{C}_{free})} > 0$. The oracle function, $\mathbb{O}_d$ of the `da_dRRT*` algorithm ensures that during the exploration phase of the algorithm, which is guaranteed to take place infinitely often, the method expands a random neighbor over the tensor product roadmap. This means that for an edge in the implicit roadmap,

$$\Pr[expand(x^{\text{near}} \rightarrow x^{\text{n}})] > 0. \quad (4)$$

Using properties of a Markov chain [15] [35] (Theorem 3), the vertices on the optimal path over the tensor product roadmap can be considered to be the states of the Markov chain, and the goal vertex to be the absorbing state. From Eq. 4, it follows that the probability of any of the edge transitions at every iteration is non-zero. For a fixed n, the probability that the Markov Process does not follow the optimal path converges to 0 as $m \rightarrow \infty$. Previous work shows that the property holds even as $n \rightarrow \infty$ [35] (Theorem 6). The implementation simultaneously searches over both $\hat{\mathbb{G}}_l = \mathbb{R}_{ls} \times \mathbb{P}_r$, and $\hat{\mathbb{G}}_r = \mathbb{R}_{sr} \times \mathbb{P}_l$.

VII. Experimental Validation

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 @ 2.70GHz processors, and 128GB of RAM using the `PRACSYS` robot simulation software [27]. 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 `RRT*`and `PRM*`. The platforms used are `Motoman SDA10F`, with a torsional `DoF`, and `Baxter` on a mobile base that can rotate and translate.

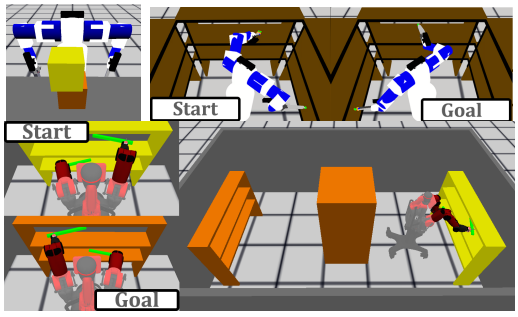

Fig. 4. *Top left* shows the tabletop setup, with the `Motoman` robot in front of a table with boxes on it. *Top center & right* show the starting and target configurations of the shelf benchmark. *Bottom left* shows the starting configuration of the mobile `Baxter` in the right shelf and the goal in the left shelf, while holding two long objects. *Bottom right* shows the walled scene between the two shelves.

For the `PRM*` algorithm and all benchmarks, 20 randomly seeded roadmaps with $50,000$ nodes are constructed in $\mathbb{C}_{full}$ and data are gathered from 20 experiments. A $50,000$ node roadmap has $\approx$ 1 million edges, and takes $\approx$ 7 hours to construct in these high dimensional spaces. Larger roadmaps run into memory scalability issues. These roadmaps in the full space occupied $\approx$ 50MB. In comparison, the space requirement for two arm roadmaps were $<$ 1MB.

For all benchmarks, both `RRT*`and `da_dRRT*`were allowed to run for $100,000$ iterations. `RRT*`is ran in 20 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. The smallest sized constituent roadmap used in da_dRRT* of 100 nodes, yields a $10,000$ vertex implicit tensor product search space, while two 1000 node roadmaps correspond an implicit 1 million vertex structure. Therein lies the power of the proposed method to deal with difficult problems in these high dimensional spaces.

The result show that the proposed algorithm provides an initial solution quickly and then converges on to better solutions. The quality of the converged solutions are better than those that exist in the PRM* roadmap constructed in $\mathbb{C}_{full}$, since the tensor roadmap is a much denser structure.

Motoman Tabletop Benchmark: A set of 20 random collision-free starts and goals are selected in the tabletop environment, shown in Fig. 4(*top-left*). 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. 5 (*left*). 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 5(*right*).

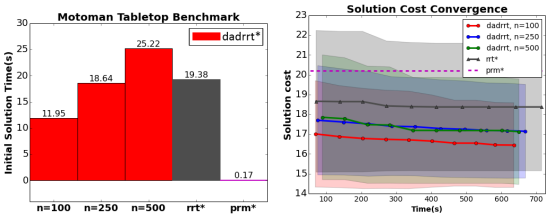

Fig. 5. Motoman Tabletop Benchmark: *Left*: The initial solution times are reported for every algorithm. *Right*: The average solution costs over time are reported. The dashed horizontal line denotes the average solution cost discovered by PRM*. The shaded regions represent the corresponding algorithm's standard deviation of cost, at that time.

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. 4 *top right*).

This is a significantly harder problem, and RRT* suffers in terms of success ratio (Fig. 6, *top left*). RRT* takes much longer to find the initial solution, as indicated by (Fig. 6 (*top right*). 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. 6 (*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 4 *bottom left*). The robot has to navigate across a cramped, walled room, to a placing configuration inside a shelf on the other side of the room.

This proves to be the most challenging problem among the three benchmarks. As shown in Fig. 7(*top left*), 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 $\mathbb{C}_{full}$ to solve harder problems. The problem is solved when a dense implicit structure, with $n = 1000$ is explored by da_dRRT*. Fig. 7 (*top right*) shows that smaller roadmaps also prove more difficult and time-consuming to search in constraining environments. This can be explained by the fact that the projection roadmap edges being evaluated would mostly be in collision. Fig. 7(*bottom*) shows that da_dRRT* finds better initial and converged solutions when compared to the instances in which PRM* succeeded.

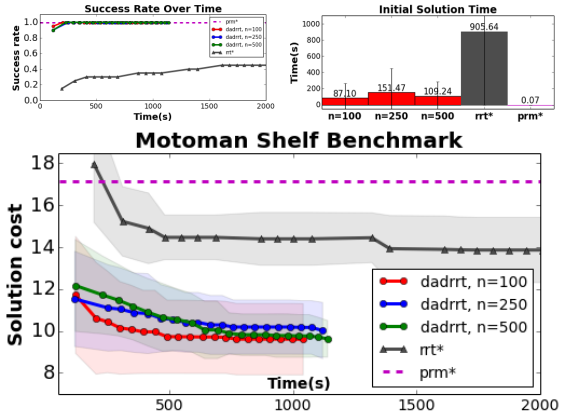

Fig. 6. Motoman Shelf Benchmark: *Top left:* Success ratios of the algorithms are shown over time. *Top right:* The initial solution times are reported for every algorithm. *Bottom:* The average solution costs over time are reported. The dashed horizontal line denotes the average solution cost discovered by PRM*. The shaded regions represent the corresponding algorithm's standard deviation of cost, at that time.

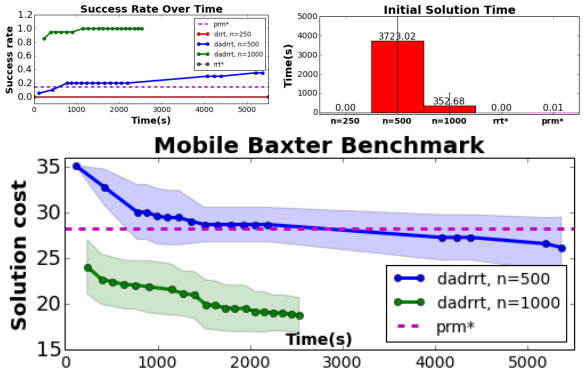

Fig. 7. Motoman Shelf Benchmark: *Top left:* Success ratios of the algorithms are shown over time. *Top right:* The initial solution times are reported for every algorithm. *Bottom:* The average solution costs over time are reported. The dashed horizontal line denotes the average solution cost discovered by PRM*. The shaded regions represent the corresponding algorithm's standard deviation of cost, at that time.

VIII. Discussion

There is a variety of applications, from warehouse automation [6] to service robotics, which can benefit from efficient operation of dual-arm robots. The proposed algorithm takes advantage of a natural decomposition of a dual-arm robot's kinematic chain. Assuming that AO roadmaps are built for these constituents spaces of the overall platform, the method can provide asymptotic optimality for the entire system, while being practically scalable.

The performance of the method can potentially be further improved through integration with recent work [31], which has provided computational benefits when planning for articulated systems by integrating information from multiple tiled roadmaps. It is interesting to consider the properties of the method after finite amount of computation time [9]. Future work can extend the results to the case of closed-chain cooperative manipulation challenges using two arms. Even though the principles should work on more than two arms, work needs to be done to explore practical ways to apply it to more complex kinematic chains.

REFERENCES

[1] M. Bonilla, L. Pallottino, and A. Bicchi. Noninteracting constrained motion planning and control for robot manipulators. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[2] F. Caccavale and M. Uchiyama. Cooperative Manipulators. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, pages 701–718. Springer, 2008.

[3] J. Canny. *The complexity of robot motion planning*. MIT press, 1988.

[4] B. Cohen, S. Chitta, and M. Likhachev. Single-and dual-arm motion planning with heuristic search. *The International Journal of Robotics Research*, 33(2):305–320, 2014.

[5] B. Cohen, M. Phillips, and M. Likhachev. Planning single-arm manipulations with n-arm robots. In *Eighth Annual Symposium on Combinatorial Search*, 2015.

[6] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. Romano, and P. Wurman. Analysis and Observations from the First Amazon Picking Challenge. *IEEE Trans. on Automation Science and Engineering*, (99):1–17, 2016.

[7] J. Cortés and T. Siméon. Sampling-based Motion Planning under Kinematic Loop Closure Constraints. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2004.

[8] A. Dobson and K. E. Bekris. Planning Representations and Algorithms for Prehensile Multi-Arm Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[9] A. Dobson, G. Moustakides, and K. E. Bekris. Geometric Probability Results For Bounding Path Quality In Sampling-Based Roadmaps After Finite Computation. In *IEEE Inter. Conf. on Robotics and Automation (ICRA)*, 2015.

[10] A. Dobson, K. Solovey, R. Shome, D. Halperin, and K. E. Bekris. Scalable Asymptotically-Optimal Multi-Robot Motion Planning. In *IEEE Intern. Symp. on Multi-Robot Systems (MRS)*, 2017.

[11] M. Dogar, A. Spielberg, S. Baker, and D. Rus. Multi-robot Grasp Planning for Sequential Assembly Operations. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[12] M. Gharbi, J. Cortés, and T. Siméon. Roadmap Composition for Multi-Arm Systems Path Planning. In *IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[13] F. Gravot and R. Alami. A Method for Handling Multiple Roadmaps and Its Use for Complex Manipulation Planning. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 3, pages 2914–2919, 2003.

[14] F. Gravot, R. Alami, and T. Siméon. Playing with several roadmaps to solve manipulation problems. In *IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems (IROS)*, volume 3, pages 2311–2316, 2002.

[15] C. Grinstead and J. Snell. *Introduction to Probability*. American Mathmatical Society, Providence, RI, 2012.

[16] K. Harada, T. Tsuji, and J.-P. Laumond. A Manipulation Motion Planner for Dual-Arm Industrial Manipulators. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 928–934, 2014.

[17] L. Janson, A. Schmerling, A. Clark, and M. Pavone. Fast Marching Tree: a Fast marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions. *International Journal of Robotics Research (IJRR)*, 34(7):883–921, 2015.

[18] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM (JACM)*, 24(1):1–13, 1977.

[19] S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research (IJRR)*, 30(7):846–894, June 2011.

[20] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[21] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. Journal of Robotics Research (IJRR)*, 5(1):90–98, 1986.

[22] A. Kimmel and K. E. Bekris. Scheduling pick-and-place tasks for dual-arm manipulators using incremental search on coordination diagrams.

[23] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 945–952. IEEE, 1994.

[24] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake. Optimization-based Locomotion Planning, Estimation and Control Design for the Atlas Humanoid Robot. *Autonomous Robots*, 40(3):429–455, 2016.

[25] S. M. LaValle and J. J. Kuffner. Randomized Kinodynamic Planning. *Intern. Journal of Robotics Research (IJRR)*, 20:378–400, May 2001.

[26] S. Leroy, J. Laumond, and Siméon. Multiple Path Coordination for Mobile Robots: A Geometric Algorithm. Barcelona, Catalonia, Spain, 1999. International Joint Conference on Artificial Intelligence (IJCAI).

[27] Z. Littlefield, A. Krontirs, A. Kimmel, A. Dobson, R. Shome, and K. E. Bekris. An Extensible Software Architecture for Composing Motion and Task Planners. In *Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, 2014.

[28] S. S. Mirrazavi Salehian, N. B. Figueroa Fernandez, and A. Billard. Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty. In *2016 Robotics: Science and Systems Conference*, number EPFL-CONF-218480, 2016.

[29] P. A. O'Donnell and T. Lozano-Pérez. Deadlock-free and Collision-free Coordination of Two Robot Manipulators. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 484–489, 1989.

[30] M. J. Pelling. Formulae for the Arc-Length of a Curve in R^N. In R. A. Brualdi, editor, *The American Mathematical Monthly*, volume 84, pages 465–467. 1977.

[31] O. Salzman, K. Solovey, and D. Halperin. Motion planning for multilink robots by implicit configuration-space tiling. *IEEE Robotics and Automation Letters*, 1(2):760–767, 2016.

[32] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: III. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Barriers. *The International Journal of Robotics Research*, 2(3):46–75, 1983.

[33] T. Siméon, S. Leroy, and J.-P. Laumond. Path Coordination for Multiple Mobile Robots: A resolution-complete algorithm. *IEEE Transactions on Robotics and Automation*, 18(1):42–49, 2002.

[34] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic. Dual-arm Manipulation: A Survey. *Robotics and Autonomous Systems*, 60(10):1340–1353, 2012.

[35] K. Solovey, O. Salzman, and D. Halperin. Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *Int. Journal of Robotics Research (IJRR)*, 35(5):501–513, 2016.

[36] P. Svestka and M. Overmars. Coordinated Path Planning for Multiple Robots. *Robotics and Autonomous Systems*, 23:125–152, 1998.

[37] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos. Nonholonomic navigation and control of cooperating mobile manipulators. *IEEE Transactions on robotics and automation*, 19(1):53–64, 2003.

[38] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann. Humanoid motion planning for dual-arm manipulation and re-grasping tasks. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2464–2470. IEEE, 2009.

[39] N. Vahrenkamp, E. Kuhn, T. Asfour, and R. Dillmann. Planning multi-robot grasping motions. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 593–600. IEEE, 2010.

[40] G. Wagner and H. Choset. Subdimensional Expansion for Multirobot Path Planning. *Artificial Intelligence Journal*, 219:1024, 2015.

[41] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. A. Bagnell, and S. Srinivasa. CHOMP: Covariant Hamiltonian Optimization for Motion Planning. *International Journal of Robotics Research (IJRR)*, 2013.

[42] R. Zurawaski and S. Phang. Path Planning for Robot Arms operating in a Common Workspace. In *Int. Conf. on Power Electronics and Motion Control*, pages 618–623, 1992.