IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Generating and Optimizing Topologically Distinct Guesses for Mobile Manipulator Path Planning with Path Constraints

Rufus Cheuk Yin Wong, Mayank Sewlia, Adrian Wiltz, Dimos V. Dimarogonas

***Abstract*—Optimal path planning is prone to convergence to local, rather than global, optima. This is often the case for mobile manipulators due to nonconvexities induced by obstacles, robot kinematics and constraints. This paper focuses on planning under end effector path constraints and attempts to circumvent the issue of converging to a local optimum. We propose a pipeline that first discovers multiple homotopically distinct paths, and then optimizes them to obtain multiple distinct local optima. The best out of these distinct local optima is likely to be close to the global optimum. We demonstrate the effectiveness of our pipeline in the optimal path planning of mobile manipulators in the presence of path and obstacle constraints.**

***Index Terms*—Mobile Manipulation, Motion and Path Planning, Constrained Motion Planning, Optimization and Optimal Control**

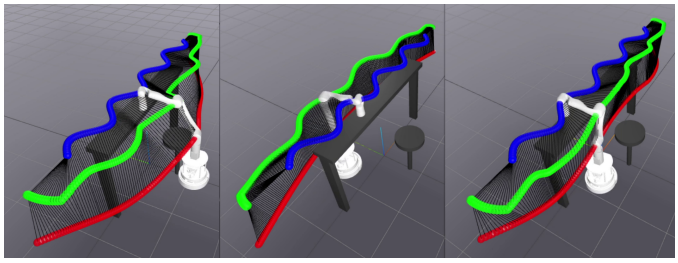


Figure 1: Mobile manipulator executing three homotopically distinct locally optimal paths given a desired end effector path. Blue shows the desired end effector path, green and red shows the computed elbow and base paths respectively.

I. Introduction

OPTIMAL path planning for mobile manipulators is commonly done by formulating and solving a nonlinear program (NLP) using gradient-based optimization approaches. One major challenge with this approach is that often the constraints introduced to the planning problem, such as obstacle avoidance, end effector path constraints, cause the NLP to be highly nonconvex. This causes gradient based optimization approaches to only solve them to local optimality. While solving nonconvex NLPs to global optimality in general is NP-hard, one potential mitigation is to generate multiple distinct local optima and choose the best among them. This increases the likelihood of actually finding the global optimum. In the sequel, we denote the optimum among multiple distinct local optima a *multi-local optimum*.

The challenge of obtaining a multi-locally optimal path is computing multiple distinct local optima since most research has only been conducted on finding a single local optimum [1], [2]. Using the observation that the local optimum returned by gradient-based optimization approaches usually stays within the same homotopy class as the provided initial guess [3], we propose a pipeline that first discovers homotopically distinct paths and then uses them as initial guesses for an NLP. This allows for generating multiple distinct local optima, and subsequently finding the *multi-local optimum*.

We apply our pipeline to the path planning of mobile manipulators consisting of a 6-revolute (6R) elbow manipulator attached to a nonholonomic differential drive base. We further require that the end effector follows a predetermined path. Such end-effector path constraints arise naturally in applications such as painting, welding or wiping a table.

The contribution of this paper is the development of a path planning pipeline for mobile manipulators under end effector path constraints and produces a *multi-locally optimal* solution. To this end, we propose a method for generating a low dimensional configuration graph to be used with the Neighborhood Augmented Graph Search (NAGS) algorithm [4]. Additionally, we propose several modifications to the NAGS algorithm that enhances its accuracy. Furthermore, an NLP is formulated to produce distinct locally optimal paths from the guesses provided by the modified NAGS algorithm. Finally, the effectiveness of the pipeline is demonstrated with simulation results along with a comparison study with existing methodologies.

The remainder is organized as follows. In Section II, related work is reviewed, and in Section III, the problem under consideration is stated. Section IV presents the proposed planning pipeline in detail, and Section V presents some experimental results demonstrating the efficacy of our pipeline. Section VI offers a conclusion with some discussion.

II. Related Work

A. Constrained Motion Planning

Motion planning in high dimensional space such as mobile manipulators under end effector constraints has been well studied. Many current methods build upon the rapidly exploring random tree (RRT) [5] and encode the constraints geometrically during tree construction [6]–[9]. These have been generalized and incorporated into the Implicit Manifold

Manuscript received: April, 25, 2025; Revised August, 23, 2025; Accepted September, 27, 2025.

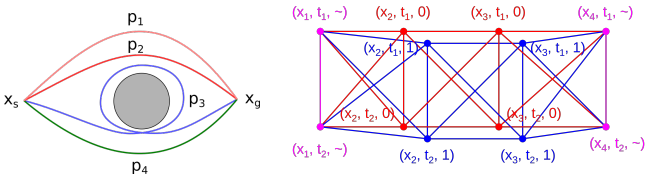
This paper was recommended for publication by Editor Aniket Bera upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by the Swedish Research Council, the Knut and Alice Wallenberg Foundation, and the Swedish Foundation for Strategic Research.

The authors are with the Division of Decision and Control Systems, School of EECS, Royal Institute of Technology (KTH), 100 44 Stockholm, Sweden [rcywong,sewlia,wiltz,dimos]@kth.se

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) (b)

Figure 2: (a) Given the grey obstacle, p_1 and p_2 belong to the same $\mathcal{H}$-class (homotopically equivalent) while p_2, p_3, p_4 each belong to a different $\mathcal{H}$-class (homotopically distinct). (b) Illustration of a 2D cross section (y axis omitted) of the configuration graph.

Configuration Space (IMACS) framework [10], decoupling the planning algorithm from the constraint adherence. These sampling approaches can often be postprocessed to produce a locally optimal path. However, the randomized nature of RRT-based algorithms implies that there is relatively little control over the characteristics of the paths returned, e.g. the topological properties.

B. Optimal Path Planning

Trajectory optimization is a commonly used technique in optimal path planning. This involves formulating the path-finding problem as a mathematical program with costs and constraints, which is then solved with an optimizer. This field is well studied with many successful algorithms such as CHOMP [1] and TrajOpt [2]. Both of these approaches use a direct transcription based technique [11], which involves discretizing the trajectory into a fixed number of discrete samples. These approaches generally scale well with the number of decision variables and constraints. However, the presence of nonconvex constraints and cost functions leads to results that are locally, not globally optimal, and heavily dependent on the initial guess provided.

C. Topological Path Planning

Topological path planning focuses on finding and quantifying paths based on their topological features. Often, the feature of interest is a path's homotopy class ($\mathcal{H}$-class) within a robot's configuration space. Paths of different $\mathcal{H}$-class cannot be smoothly deformed into each other without colliding with obstacles (Fig. 2a). Many probabilistic methods for finding homotopically distinct paths have been proposed [12]–[14]. However, they generally scale poorly to high dimensions.

As such, using a lower dimensional or simpler topological path planning setup for coarse global planning followed by optimal path planning approaches for local refinement is a common approach to combine the best of both worlds. This pipeline is effective in generating optimal trajectories for mobile ground robots [15], [16], quadrotors [17] and manipulators [18], [19].

For applying this topological and optimal path planning pipeline to mobile manipulators, a major problem is the determination of the $\mathcal{H}$-class, which is very challenging for high-dimensional configuration spaces [4]. A novel Neighborhood Augmented Graph Search (NAGS) algorithm [4] has recently been proposed that allows finding topologically distinct paths in higher dimensions. In our work, we leverage a modified version of NAGS to identify homotopically distinct paths and use optimal path planning for the local refinement.

III. Problem Formulation

The path planning problem concerns a 6-degree-of-freedom (DoF) fully actuated elbow manipulator attached to a non-holonomic differential drive mobile base. The manipulator is assumed to have a 3DoF spherical wrist that handle any end effector orientation constraints. This is the case for most mobile manipulators available today. As such, we refer to the wrist as the end effector and only consider its position.

The base is characterized by its position $x_b = [x, y]^T \in \mathbb{R}^2$ and orientation $\theta \in S^1$. The base motion is governed by

$$\dot{x}_b = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} u_1, \quad \dot{\theta} = u_2 \tag{1}$$

where $u_1, u_2 \in \mathbb{R}$ are control inputs.

As opposed to the more common approach of describing the arm in joint angle coordinates, we instead choose to express the arm in maximal coordinates [11]. In maximal coordinates, links are described by their position in space. This allows for a more natural incorporation of the end effector constraint that we will exploit in Section IV-A. Define $x_{b\perp} = [x_b, 0]^T = [x, y, 0]^T$. Given the base position x_b, the arm can be characterized by its elbow position $x_w \in \mathbb{R}^3$ and end effector position $x_e \in \mathbb{R}^3$, both in world cartesian coordinates. Let l_1 be the upperarm length and l_2 be the forearm length. The elbow and end effector positions are subject to the following kinematic constraints:

$$\begin{aligned} \|x_w - x_{b\perp}\|_2 &= l_1 \\ \|x_w - x_e\|_2 &= l_2 \\ \exists a, b \in \mathbb{R} : x_w - x_{b\perp} &= a(x_e - x_{b\perp}) + \begin{bmatrix} 0 & 0 & b \end{bmatrix}^T \end{aligned} \tag{2}$$

The last constraint states that base, elbow and end effector positions projected to the xy-plane are collinear. This reflects the fact that the upperarm and elbow cannot roll. The dynamics of the arm is given by

$$\dot{x}_w = u_3, \quad \dot{x}_e = u_4 \tag{3}$$

where $u_3, u_4 \in \mathbb{R}^3$ are control inputs, subject to the constraints in Eq. (2). The robot configuration q is then fully defined by

$$q = \begin{bmatrix} x_b^T & \theta & x_w^T & x_e^T \end{bmatrix}^T$$

subject to the aforementioned constraints. Note that this approach of using maximal coordinates to encode end effector constraints is not specific to the 6-DoF elbow manipulator and can be applied to arms with different kinematics by adjusting Eq. (2) and (3), correspondingly [20].

Obstacles are assumed to be defined via an obstacle function $\text{obs}(q)$ which returns *True* if and only if the given robot configuration q is colliding with an obstacle.

A desired end effector path is given in the form of a function $x_e(k)$ with $k \in [0, 1]$ being the normalized path parameter. The planning problem, then is to find a feasible path, satisfying the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

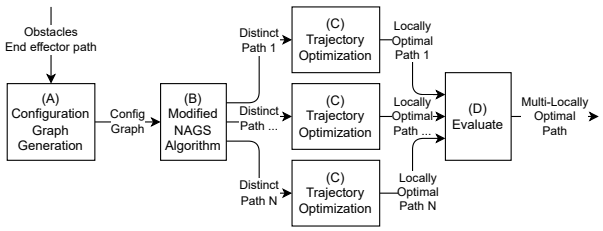


Figure 3: The planning pipeline

Algorithm 1 Pipeline for finding multi-locally optimal paths

Input:

x_e: $[0,1] \to \mathbb{R}^3$: Desired end effector path
n: Number of distinct local optima to evaluate
dt: Optimization timestep interval
T: Number of optimization timesteps

Output: $Q^\star = \{q(t_0), q(t_1), \ldots, q(t_T)\}$, $t_0 = 0$, $t_T = 1$

1: **function** FINDPATH(x_e, n, dt, T)
2: $G = (V, E) :=$ ConfigurationGraphGeneration(x_e)
3: $\begin{bmatrix} (x_{b1}, x_{w1}, t_1) \\ (x_{b2}, x_{w2}, t_2) \\ \vdots \\ (x_{bn}, x_{wn}, t_n) \end{bmatrix} :=$ modifiedNAGS(G, n)
4: **for all** $i \in 1 \ldots n$ **do** ▷ can be run in parallel
5: $(\text{cost}_i, Q_i^\star) :=$ TrajOpt($x_{bi}, x_{wi}, t_i, dt, T$)
6: $i^\star := \arg\min_i(\text{cost}_i)$
7: **return** $Q_{i^\star}^\star$

kinematic and end effector path constraints, and which does not collide with obstacles, minimizing

$$\int_0^1 \|\mathbf{u}(t)\|_2 dt \tag{4}$$

IV. Methodology

The proposed planning pipeline consists of four main steps, illustrated in Fig. 3 and Algorithm 1.

(A) First (line 2), we generate the collision-free configuration space graph (CG) representing the valid robot configurations and transitions.
(B) Then (line 3), we apply a modified NAGS algorithm, adapted from [4], which takes as input the CG and finds a pre-specified number of homotopically distinct paths within the graph.
(C) Next (line 4-5), the homotopically distinct paths are used as initial guesses and the trajectory optimization problem is solved for each guess.
(D) Finally (line 6-7), we compare the optimized paths from the different initial guesses and choose the best path.

A. Configuration Graph Generation

The goal of this step is to transform the constrained high-dimensional continuous space of collision-free robot configurations into an unconstrained low-dimensional discrete graph for the subsequent NAGS algorithm. The base heading and nonholonomic constraints are ignored at this stage. The dimensionality reduction is achieved by a change of coordinates from $[x_b, x_w, x_e]^T \in \mathbb{R}^2 \times \mathbb{R}^3 \times \mathbb{R}^3$ to $[x_b, k, w]^T \in \mathbb{R}^2 \times [0,1] \times \{0,1\}$ by noticing that x_e is fully defined by the path parameter k and that given x_b and x_e, there only exists two feasible elbow positions: elbow up and elbow down, represented by $w = 1$ and $w = 0$ respectively, with $w \in \{0,1\}$. This is a direct results from Eq. (2). This parametrization reduces the configuration space dimensionality allowing for a simpler configuration graph and thus better runtime performance. In the remainder of this section, we abuse notation and use $[x, y, k, w]^T$ and $[x_b, x_w, x_e]^T$ interchangeably with the understanding that the former can always be mapped to the latter via standard IK procedures [21].

The configuration graph (CG) is given by $G = (V, E)$ where V is the set of vertices and E is the set of undirected edges. Beginning with V, vertices are defined via a discretization of the configuration space $(x, y, t, w) \in \mathbb{R}^2 \times [0,1] \times \{0,1\}$ by predefined discretization intervals $\Delta x, \Delta y, \Delta t$. We define the discretized bounded configuration space as

$$\begin{aligned} C := &\{x_{\min}, x_{\min} + \Delta x, \ldots, x_{\max}\} \\ &\times \{y_{\min}, y_{\min} + \Delta y, \ldots, y_{\max}\} \\ &\times \{t : 0, \Delta t, \ldots, 1\} \times \{0,1\} \end{aligned}$$

Define $C_{\text{free}} \subseteq C$ to be the configurations not in collision with obstacles. Furthermore, given the upperarm link length l_1 and forearm link length l_2, the distance between the base and end effector cannot be greater than the full arm length $(l_1 + l_2)$, thus we define

$$C_{\text{kinematic}} = \{(x, y, t, w) : \|[x, y, 0]^T - x_e(t)\|_2 \leq l_1 + l_2\}$$

The set of vertices is then given by $V = C_{\text{free}} \cap C_{\text{kinematic}}$.

The vertices within an elbow configuration w are connected by collision-free edges in a grid-like fashion with diagonals for coordinates x, y, t. Transitions between elbow configurations w can only occur at the joint singularity, when $\|[x, y, 0]^T - x_e(t)\|_2 = l_1 + l_2$. A simplified example is illustrated in Fig. 2b. The edge cost is defined as the Euclidean distance between vertices in the (x, y, t) coordinates.

The construction of a configuration graph for general mobile manipulators follows similarly by solving for each joint's possible cartesian positions given x_b and x_e, according to their kinematic constraints in Eq. (2).

B. Modified Neighborhood Augmented Graph Search

The next step is to generate topologically distinct guesses. Our approach is based on a modified version of the NAGS algorithm [4, Algorithm 1]. The original algorithm along with our modifications colored in blue, orange and magenta, is presented in Algorithm 2.

The main idea behind the original NAGS algorithm is to include approximations of the path tangents to vertices in Dijkstra's Algorithm [22]. This is done by using a vertex's path neighborhood set (PNS) [4, Algorithm 3], which is computed by running a reverse A* search [23] on the graph for a fixed search depth r from the current vertex back to the starting vertex. Since homotopically distinct paths terminating at the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Algorithm 2 Modified NAGS Algorithm

Input:
$q_s \in V$: Start configuration
$q_g \in V$: Goal configuration
$\mathcal{N}_G$: Neighbor/successor function for graph G
$\mathcal{C}_G : V \times V \to \mathbb{R}^+$: Cost function
n_{req}: Required number of homotopically distinct paths
@computePS: $V \times (V_N, E_N)$: parent set computation
Output: G_N: Graph with costs and parent set for every vertex

function SEARCHNAG(q_s, q_g, $\mathcal{N}_G$, $\mathcal{C}_G$, n_{req})
  $V_N := \{v_s\}, \quad E_N := \emptyset$
  $v_s := (q_s, \{q_s\}), \quad g(v_s) := 0$
  $Q := \{v_s\}, \quad v := v_s$
  $n := 0$
  while $Q \neq \emptyset \wedge n < n_{\text{req}}$ **do**
    $v := (q, U) = \arg\min_{v' \in Q} g(v')$
    $Q = Q - v$
    $V_N = V_N \cup \{v\}$ ▷ add vertex when visiting
    $E_N = E_N \cup \{(v, v.\text{came\_from})\}$
    ~~$U' = \text{computePNS}(v, (V_N, E_N))$~~
    $U' = \text{computePS}(v, (V_N, E_N))$
    ~~**for all** $q' \in \mathcal{N}_G(v)$ **do**~~
    for all $q' \in \mathcal{N}_G(v) : \begin{cases} \exists w \in \mathcal{N}_G(v) \\ w \equiv (q', U') \end{cases}$ **do**
      ▷ handle equivalent vertices first
      $v' := (q', U')$
      $g' = g(v) + C_G(q, q')$
      $w = v'$
      $E_N = E_N \cup \{(v, w)\}$
      if $g' < g(w) \wedge w \in Q$ **then**
        $g(w) = g'$
        $w.\text{came\_from} = v$
        $w.U = U'$
    for all remaining $q' \in \mathcal{N}_G(v)$ **do**
      $v' := (q', U')$ ▷ guaranteed new vertices
      ~~$V_N = V_N \cup \{v'\}$~~ ▷ do not add vertex here
      ~~$E_N = E_N \cup \{(v, v')\}$~~
      $g(v') = g'$
      $v'.\text{came\_from} = v$
      $Q = Q \cup \{v'\}$
      if $q' = q_g$ **then** $n = n + 1$
  return $G_N = (V_N, E_N)$

same vertex should have distinct path tangents, vertices are considered distinct if their PNS do not intersect. Note the distinction between CG vertex (vertices in the CG) and NAG vertex (vertices that are incrementally added to the NAG during the Dijkstra search). Each NAG vertex corresponds to one CG vertex; however, multiple NAG vertices may correspond to the same CG vertex. Two NAG vertices v_1 and v_2 are said to be *coincident* if they correspond to the same CG vertex. The equivalence ($\equiv$) of two coincident NAG vertices v_1, v_2 are thus defined as follows [4, Definition 4]

$$v_1 \equiv v_2 \iff v_1.\texttt{cg} = v_2.\texttt{cg} \wedge v_1.\texttt{pns} \cap v_2.\texttt{pns} \neq \emptyset$$

where $v.\texttt{cg}$ and $v.\texttt{pns}$ retrieves NAG vertex v's corresponding CG vertex and PNS, respectively. We improve upon the original NAGS algorithm to address several shortcomings:

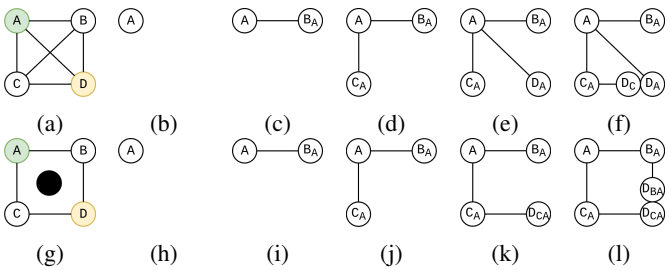


Figure 4: Top row: (a) shows a simple CG with the green start vertex and yellow goal vertex. The edge weight corresponds to the length depicted. Note that this CG only has one homotopically unique path from start to goal. (b)-(f) corresponds to the successive iterations as the NAG grows using $r = 1$. The subscript indicates the PNS of the NAG vertex. Notice in (f) that since NAG vertex D_C and D_A have disjoint PNS, NAGS incorrectly identifies them as homotopically distinct. Bottom row: (g) shows a simple CG with an obstacle in the middle. Notice that there are two homotopically distinct paths from the start to goal. (h)-(l) corresponds successive iterations with $r = 2$. Notice in (l) that the NAG vertices D_{BA} and D_{CA} have overlapping PNS, thus NAGS incorrectly identifies them as homotopically equivalent.

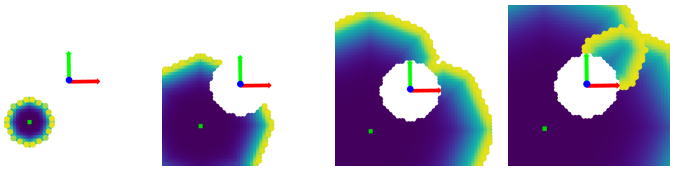


Figure 5: Successive iterations of the NAG. Notice that obstacles cause the yellow wavefront to split and then merge.

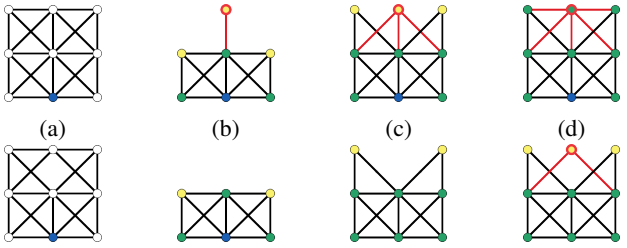
1) Tiny obstacles: The original NAGS algorithm suffers from the inability to distinguish homotopically distinct paths around tiny obstacles regardless of the value of r chosen. This is exemplified in Fig. 4. Notice that regardless of the value of r, the original NAGS algorithm is not able to correctly identify the homotopically distinct paths in the top and bottom cases simultaneously. This stems from the fact that the continuous path tangent is poorly represented in a discrete graph structure. As such, the overlap in the path tangent, approximated by the overlap of the PNS, is easily over- or underestimated, causing the incorrect identification of homotopically distinct paths.

Instead of a static approximation of the path tangent, we improve performance by using a dynamic local description of the wavefront of the open set instead (Fig. 5). The following observation can be made upon careful inspection of the wavefront as it visits a vertex. Given the current state of the NAG $G_N = (V_N, E_N)$, define the parent set $\mathcal{P}(v)$ of NAG vertex v as

$$\mathcal{P}(v) := \{v' : \exists (v, v') \in E_N\}$$

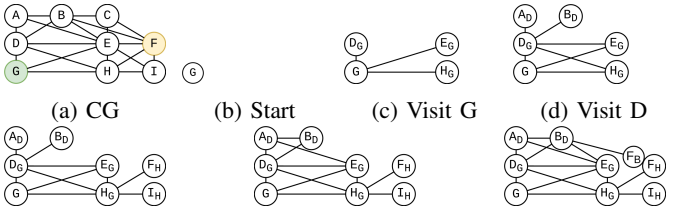
We observe that in the absence of obstacles, the parent set (PS) of a vertex starts on the shortest path and grows to adjacent

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



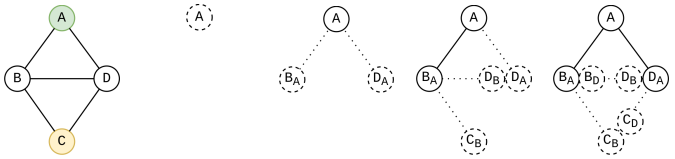
(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6: Top row: (a) shows a CG with the blue start vertex. (b)-(d) shows the successive iterations of the NAG. Yellow vertices are those in the open set (wavefront) while green vertices have already been visited. Consider the NAG vertex circled in red. The red edge connects the circled vertex to its parents. Bottom row: (e) shows a CG with an obstacle. (f)-(h) shows successive iterations of the NAG.



(a) CG (b) Start (c) Visit G (d) Visit D (e) Visit H (f) Visit A (g) Visit B

Figure 7: (a) Non-uniformly discretized CG with $GH < GD + DB < GE$. Note that there is only 1 homotopically unique path between G and F. (b)-(g) Progression of the NAGS algorithm using PS. Subscript indicates the parent of the vertex. In (g), $\mathcal{P}(F_B) = \{B_D\}$, $\mathcal{P}(F_H) = \{H_G\}$. Since $\mathcal{P}(F_B)$ is not adjacent to $\mathcal{P}(F_H)$, we have $F_B \not\equiv F_H$. Hence the algorithm incorrectly determines that there are two homotopically distinct paths from G to F.



(a) CG (b) Start (c) Visit A (d) Visit B (e) Visit D

Figure 8: (a) CG with uniform edge lengths. Note that there is only 1 homotopically unique path between A and C. (b)-(e) Progression of the NAGS algorithm using PS. Subscript indicates the parent of the vertex. Dotted vertices represent vertices in the heap. Notice that the order for visiting B_D, D_B, C_B, C_D is undefined as they all have the same path cost. Furthermore, whether or not $C_B \equiv C_D$ depends on the order in which the four vertices are visited. If C_B and C_D are visited before B_D or D_B, then $C_B \not\equiv C_D$.

vertices. In the presence of obstacles, this rule is broken. The PS no longer grows to adjacent vertices. This is demonstrated in Fig. 6. In effect, the PS acts as a local description of the wavefront.

This motivates the use of the PS to detect whether two paths are homotopically distinct or not. Two PS $\mathcal{P}_1, \mathcal{P}_2$ are said to be *adjacent* if the following holds:

$$\text{adj}_{E_N}(\mathcal{P}_1, \mathcal{P}_2) \iff \exists v_1 \in \mathcal{P}_1, v_2 \in \mathcal{P}_2 : (v_1, v_2) \in E_N$$

We then redefine the equivalence relation ($\equiv$) as follows:

Definition 4.1 (Equivalence between NAG vertices): For coincident NAG vertices v_1 and v_2 and NAG $G_N = (V_N, E_N)$,

$$v_1 \equiv v_2 \iff v_1\texttt{.cg} = v_2\texttt{.cg} \wedge \text{adj}_{E_N}(\mathcal{P}(v_1), \mathcal{P}(v_2))$$

where $v\texttt{.cg}$ retrieves the corresponding CG vertex of v.

To illustrate the effect of using PS, consider the examples in Fig. 4f again. Using PS, $\mathcal{P}(D_C) = \{C_A\}, \mathcal{P}(D_A) = \{A\}$. Since C_A is adjacent A, $D_C \equiv D_A$ and they are considered homotopically identical. For Fig. 4l, $\mathcal{P}(D_{CA}) = \{C_A\}, \mathcal{P}(D_{BA}) = \{B_A\}$. Since C_A and B_A are not adjacent, the two NAG vertices will be considered distinct, correctly identifying the two homotopically distinct paths.

2) Non-uniform discretization: In the original NAGS algorithm, r must be fine tuned to account for potentially large differences in edge weights. The PS modification mentioned in Section IV-B1 itself does not alleviate this issue. One example is illustrated in Fig. 7. This is due to the fact that the original NAGS algorithm adds a vertex to the NAG based on the parent of that vertex. We thus apply the changes in line 9-10 and line 25-26 of Algorithm 2. These modifications ensure that vertices are added to the NAG in the order of the cost to the vertex itself, rather than the parent. The effect of these modifications is that F_E will be added to the NAG before F_B. Since F_E and F_H are equivalent, the edge (E_G, F_H) is inserted, causing the PS of F_H to expand to E_G. Then, $\mathcal{P}(F_H) = \{H_G, E_G\}$ will be adjacent to $\mathcal{P}(F_B) = \{B_D\}$ and thus $F_B \equiv F_H$.

3) Ambiguous visiting order: Similar to the above, the order in which vertices are visited impacts both PS and PNS calculation. This is illustrated in Fig. 8. To tackle the nondeterministic visiting order of paths of the same length, we prioritize processing equivalent vertices first (line 13-14) before processing nonequivalent vertices (line 23). This prevents equivalent vertices being incorrectly considered distinct due to visiting order.

4) Sufficient Condition Sketch: We provide a more general sufficient condition for detecting homotopically distinct paths, only requiring obstacles that cause the removal of edges/vertices in the CG.

Proposition 1: Two locally shortest (geodesic) paths, p_1 and p_2, from the CG vertex v_s to v_g that encloses an obstacle will generate distinct NAG vertices.

Proof Sketch: Note that each obstacle causes the removal of edges/vertices in the configuration graph, creating a chordless cycle R_0. A chordless cycle is a cycle of length at least four in which no two vertices are joined by an edge outside of the cycle itself. Define R_k to be the set of vertices adjacent to but not contained in R_{k-1}.

Let $\{n_1, \ldots, n_j\}$ be the NAG vertices of p_1 and $\{m_1, \ldots, m_h\}$ be the NAG vertices of p_2. Suppose path lengths $l(p_2) \geq l(p_1)$. Note that n_1 and m_1 both correspond to the same CG vertex v_s and $n_1 \equiv m_1$ while n_j and m_h both correspond to v_g, but it is yet to be determined whether they are equivalent or not.

The inductive proof proceeds as follows:

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

1) Base condition: Consider $v_g \in R_0$. In order for p_1 and p_2 to enclose R_0 and be geodesics, we must have $n_{j-1}, m_{h-1} \in R_0$. Since R_0 is a chordless cycle, there is no edge between n_{j-1} and m_{h-1}. $\mathcal{P}(n_j) = \{n_{j-1}\}$ is not adjacent $\mathcal{P}(m_h) = \{m_{h-1}\}$. Hence $n_j \not\equiv m_h$.
2) Inductive step: Assume the proposition is true $\forall v'_g \in R_{k-1}$. We wish to show it to be true for $v_g \in R_k$. Suppose the contrary, that for some $v_g \in R_k$, $n_j \equiv m_k$. This requires that $\mathcal{P}(m_h) = \{m_{h-1}\}$ be adjacent to $\mathcal{P}(n_j) = \{n_{j-1}\}$. We consider two cases for m_{h-1}
   a) $m_{h-1} \in R_k \cup R_{k+1}$: This cannot be the case if p_1, p_2 enclose the obstacle, $l(p_2) \geq l(p_1)$ and they are locally shortest paths. This can be seen since there must exist some point $m_a \in p_2 \cap R_0$ and the subpath $(m_a, \ldots, m_{h-1})$ is locally shortest.
   b) $m_{h-1} \in R_{k-1}$: Then there exist some $v'_g = m_{h-1} \in R_{k-1}$ such that p'_1 and p'_2 are equivalent, contradicting the induction hypothesis.

   Hence the proposition must be true for $v_g \in R_k$.

Thus we show the proposition to be true for pairs of locally shortest paths. □

The generalization to all possible pairs of paths from v_s to v_g follows based on two facts: 1) If two NAG vertices n_{j-1}, m_{h-1} are adjacent, then a path via n_{j-1} to m_{h-1} must have been deemed homotopically equivalent to a path via m_{h-2} to m_{h-1} in earlier iterations. 2) Dijkstra's algorithm always finds shorter paths first. Thus, homotopic equivalence is determined by the locally shortest path.

C. Trajectory Optimization

The goal of this step is to use the results from the previous section to refine the path, considering all constraints of the original planning problem. In addition to including the base heading and the nonholonomic constraints, a finer time discretization is used to ensure that constraints are satisfied more precisely.

The trajectory optimization problem is given as

$$\min_{\substack{x_b[k], x_w[k], \theta[k] \\ u_1[k], u_2[k], \Delta x_w[k]}} \quad \sum_{k=0}^{T} \|u_1[k]\|_2^2 + \|u_2[k]\|_2^2 + \|\Delta x_w[k]\|_2^2 \tag{5a}$$

$$\text{s.t.} \quad \|x_w[k] - x_b[k]\|_2 = l_1, \tag{5b}$$

$$\|x_w[k] - x_e[k]\|_2 = l_2, \tag{5c}$$

$$x_w[k] - x_b[k] = a(x_e[k] - x_b[k]) + \begin{bmatrix} 0 \\ 0 \\ b \end{bmatrix}, \tag{5d}$$

$$x_b[k+1] = x_b[k] + \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} u_1[k] dt, \tag{5e}$$

$$\theta[k+1] = \theta[k] + u_2[k] dt, \tag{5f}$$

$$x_w[k+1] = x_w[k] + \Delta x_w[k] dt, \tag{5g}$$

$$\text{obs}(x_b[k], x_w[k], x_e[k]) = \text{False} \tag{5h}$$

The objective (5a) is to minimize the discretized cost in Eq. (4). This is subject to (5b)-(5d) which enforce the kinematic constraints in Eq. (2), and (5e)-(5g) which enforce the dynamic constraints in Eq. (1) and (3). Finally, (5h) enforces collision avoidance at each timestep.

D. Evaluate Local Optima

The final step is to compare and select the least cost path among the locally optimal paths from the previous step. Formally, given the locally optimal trajectories $Q_i^\star$ and associated cost_i for $i \in \{1, \ldots, n\}$, index $i^\star$ of the trajectory with the least cost is given by

$$i^\star := \arg\min_i (\text{cost}_i)$$

The *multi-locally optimal* path is then $Q_{i^\star}^\star$ which is the optimal path among the local optima $\{Q_1^\star, \ldots, Q_n^\star\}$.

V. Results

In this Section, we compare our method for generating topologically distinct initial guesses against sampling-based approaches and simple interpolation. For all planning problems, we generate the CG by discretizing the end effector path at 0.05m intervals. Base positions are discretized at a resolution of 0.1m. Edges in the CG are further subsampled at 0.01m for collision checking. The results of both modified NAGS and sampling-based methods are all used as initial guesses for the NLP specified in Section IV-C. The optimization problem is formulated in Drake [24] and solved with SNOPT [25] using discretization $T = 200$ and timestep $dt = 0.2$. All final costs represent optimized costs in Eq. (5a) after using the initial guess to solve the NLP. Runtime only includes the time taken to generate the the initial guesses and does not include the time for solving the NLP . The code for the implementation and comparison, as well as interactive recordings of the results can be found at https://github.com/rcywongaa/topologically_distinct_guesses. We investigate three planning scenarios.

A. Two Sphere Obstacles with Straight Line Path Constraint

Planning Problem 1 involves finding the optimal path for a simple two-link mobile manipulator in the presence of two spherical obstacles, subject to an end effector constraint in the form of a straight line.

Our algorithm is compared against two constrained sampling-based approaches: the IMACS-RRTConnect [26] algorithm (emulates CBIRRT2 [7], TB-RRT [8], AtlasRRT [9]) and the IMACS-KPIECE [27] algorithm which shows superior results in high dimensional constrained configuration space [10]. The path tolerance was set to 0.05m.

The sampling-based planners are compared in singleshot mode (1), where only one path is generated and evaluated, and multishot mode (n), where the planner keeps generating paths until at least one path belonging to each of the n homotopic classes is generated. Both modes are subject to a 5 minute timeout. The sampling-based approaches were set up using OMPL [28] and MoveIt2 [29] [30]. Additionally, a simple IK-based interpolation method is also compared.

The results are summarized in Table I, averaged over 10 trials. It can be seen that our algorithm can generate topologically distinct initial guesses more quickly as indicated by the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Method	Planning Problem 1 (n=4)			Planning Problem 2 (n=3)		
	Runtime(s)	NLP success rate	Final cost	Runtime(s)	NLP success rate	Final cost
CG+Modified NAGS (n)	1.02+0.49	1.0 / 1.0 / 1.0 / 1.0	**2.33** / 6.02 / 6.09 / 6.05	0.90+2.35	1.0 / 1.0 / 1.0	**2.32** / 2.98 / 2.99
IMACS-KPIECE (1)	1.79	0.3	14.37	timeout	*	*
IMACS-KPIECE (n)	29.9	0.6 / 0.2 / 0.2 / 0.1	10.57 / **5.75** / 25.64 / 10.03	timeout	* / * / *	* / * / *
IMACS-RRTConnect (1)	37.3	0.3	4.62	timeout	*	*
IMACS-RRTConnect (n)	timeout	* / * / * / *	* / * / * / *	timeout	* / * / *	* / * / *
Simple Interpolation (1)	0.01	0.0	*	0.01	0	*

Table I: Comparison between different initialization methods for the NLP. * indicates failure of current or prior steps. (1) indicates singleshot mode. (n) indicates multishot mode, where n is the number of $\mathcal{H}$-classes required. The values separated by "/" are data corresponding to paths of each $\mathcal{H}$-class. The bold text indicates the multi-locally optimal path

lower runtime, as well as produce guesses of higher quality, as indicated by the higher NLP success rate and lower final cost. Examples from modified NAGS and IMACS-KPIECE are shown in Fig. 10 and Fig. 11 respectively.

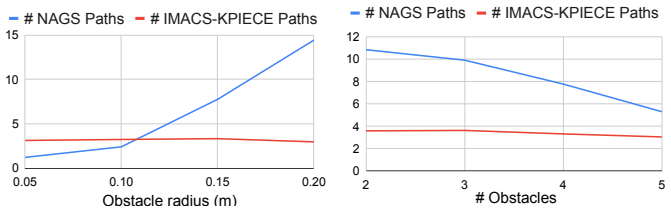
Note that the nonholonomic constraints of the mobile manipulator are only enforced in the NLP, further introducing more local optima to the optimization landscape. This highlights the importance of providing high-quality guesses to the optimizer and explains the apparent differences in final cost of trajectories belonging to the same $\mathcal{H}$-classes. Since the NAGS algorithm produces shortest paths within the $\mathcal{H}$-class, it naturally constitutes an initial guess that is closer to the global optimum, even with the added nonholonomic constraints. Additionally worth pointing out, the single-shot performance of IMACS-KPIECE is comparable to that of our CG+Modified NAGS pipeline, while its multi-shot performance is significantly worse. This indicates that the slowdown stems from the absence of homotopy awareness in IMACS-KPIECE. Since the discovery of homotopically distinct paths is inherently probabilistic, this largely explains the increased planning time.

B. Simulated Bar Table Cleaning

Planning Problem 2 involves a more realistic table cleaning scenario in which a mobile manipulator in the form of a Kinova Gen3 robot arm attached to a Turtlebot 4 is tasked with cleaning a counter table with a sine wave motion while avoiding collisions. The sampling-based planners are set up and evaluated in the same fashion as in Section V-A and the results are summarized in Table I, also averaged over 10 trials. The final results are shown in Fig. 1. Particularly, note that the highly nonconvex table and chair means that conventional approaches of projecting obstacles to the ground plane and splitting base and arm motion planning would yield poor results since naive projection would either severely overestimate or underestimate the size of the obstacles. Indeed, if we project the table and chair to the ground plane and consider the mobile base inflation radius, the path where the base moves between the table and the chair would not be feasible. This planning problem also highlights the limitation of sampling based planners which struggle with narrow passages generated by the large obstacle and end effector constraints [10].

C. Randomized Tests

This test studies the effect of the number and size of the obstacles on the performance of our modified NAGS algorithm.



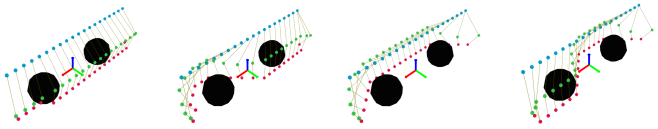
(a) (b)

Figure 9: Effect of varying number of obstacles and obstacle radius on number of paths found in 10s. Note that paths found by IMACS-KPIECE may not be homotopically distinct. (a) uses 4 obstacles. (b) uses obstacles of radii 0.15m.

The setup is similar to Planning Problem 1 with a varying number and radii of spherical obstacles. In this experiment, we focus on comparing against IMACS-KPIECE since it has been shown to be superior in performance in Planning Problem 1. Since it is difficult to predetermine the number of $\mathcal{H}$-classes in a randomized setting, we instead impose a 10s time limit for both modified NAGS and IMACS-KPIECE to generate as many paths as possible (homotopically distinct or not). The experiment is conducted with randomized obstacle positions averaged over 50 trials per setting. The results are shown in Fig. 9. It can be seen that modified NAGS outperforms IMACS-KPIECE in situations with a small number of large obstacles. Large obstacles are favorable since they lead to fewer vertices in the CG, and thus to a faster search. For a high number of obstacles, the number of homotopically distinct paths of a certain length grows rapidly. This causes the open set to grow quickly, which impacts the performance. Furthermore, since each obstacle creates infinitely many homotopically distinct paths corresponding to increasing winding numbers, there is no upper bound to the size of the open set.

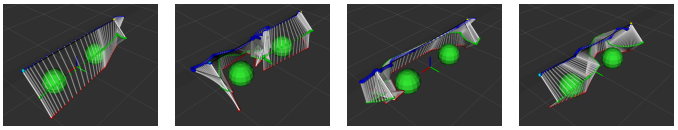
Using 4 obstacles of radius of 0.15, we repeat the experiment with the trajectory optimization step included. We additionally randomize the initial and goal headings. The respective multi-locally optimal paths are then compared. Across 50 trials, modified NAGS and IMACS-KPIECE produced at least one admissible initial guesses for 88% and 60% of the trials, respectively, where an initial guess is admissible if it allows the NLP to be solved. On average, the multi-locally optimal paths generated by the NAGS algorithm were 30.6% shorter than those generated by IMACS-KPIECE. Furthermore, 34% of the multi-locally optimal paths from modified NAGS were *not* from the first (shortest) path returned. Since

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) $\mathcal{H}$-class 1 (b) $\mathcal{H}$-class 2 (c) $\mathcal{H}$-class 3 (d) $\mathcal{H}$-class 4

Figure 10: Results of NAGS belonging to different $\mathcal{H}$-classes for Planning Problem 1



(a) $\mathcal{H}$-class 1 (b) $\mathcal{H}$-class 2 (c) $\mathcal{H}$-class 3 (d) $\mathcal{H}$-class 4

Figure 11: Results of IMACS-KPIECE belonging to different $\mathcal{H}$-classes for Planning Problem 1

the first path returned by modified NAGS is equivalent to the result of running standard Dijkstra's Algorithm [22], this also shows that modified NAGS produced better initial guesses than standard Dijkstra in those cases.

VI. Conclusion & Discussions

This paper presents a pipeline for mobile manipulator path planning under end effector path constraints that achieve multi-local optimality. Several modifications were proposed to the core NAGS algorithm enabling it to reliably distinguish homotopically distinct paths. Our algorithm performs particularly well in scenarios where kinematic structure and constraints reduce the dimensionality of the problem, and where a small number of large obstacles lead to a more compact configuration graph. In such cases, the algorithm's ability to handle large obstacles becomes especially beneficial, as these environments often introduce challenging local optima that our approach is well-equipped to address. This can be seen as a complement to sampling-based approaches, which generally work well in the absence of constraints and with smaller, more numerous obstacles. Future work may investigate ways to alleviate the curse of dimensionality when applying our algorithm to mobile manipulators with many DoFs.

References

[1] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International journal of robotics research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[2] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[4] A. Sahin and S. Bhattacharya, "Topo-geometrically distinct path computation using neighborhood-augmented graph, and its application to path planning for a tethered robot in 3-d," *IEEE Transactions on Robotics*, vol. 41, pp. 20–41, 2025.

[5] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[6] G. Oriolo and C. Mongillo, "Motion planning for mobile manipulators along given end-effector paths," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 2154–2160.

[7] D. Berenson, *Constrained manipulation planning*. Carnegie Mellon University, 2011.

[8] B. Kim, T. T. Um, C. Suh, and F. C. Park, "Tangent bundle rrt: A randomized algorithm for constrained motion planning," *Robotica*, vol. 34, no. 1, pp. 202–225, 2016.

[9] L. Jaillet and J. M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *IEEE Transactions on Robotics*, vol. 29, no. 1, pp. 105–117, 2012.

[10] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1151–1178, 2019.

[11] R. Tedrake, *Underactuated Robotics*, 2023. [Online]. Available: https://underactuated.csail.mit.edu

[12] F. T. Pokorny, D. Kragic, L. E. Kavraki, and K. Goldberg, "High-dimensional winding-augmented motion planning with 2d topological task projections and persistent homology," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 24–31.

[13] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, pp. 273–290, 2012.

[14] L. Jaillet and T. Siméon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1175–1188, 2008.

[15] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[16] W. He, Y. Huang, J. Wang, and S. Zeng, "Homotopy method for optimal motion planning with homotopy class constraints," *IEEE Control Systems Letters*, vol. 7, pp. 1045–1050, 2022.

[17] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[18] J. J. Rice and J. M. Schimmels, "Multi-homotopy class optimal path planning for manipulation with one degree of redundancy," *Mechanism and Machine Theory*, vol. 149, p. 103834, 2020.

[19] M. S. Saleem, R. Sood, S. Onodera, R. Arora, H. Kanazawa, and M. Likhachev, "Search-based path planning for a high dimensional manipulator in cluttered environments using optimization-based primitives," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8301–8308.

[20] J. Brüdigam, S. Sosnowski, Z. Manchester, and S. Hirche, "Variational integrators and graph-based solvers for multibody dynamics in maximal coordinates," *Multibody System Dynamics*, vol. 61, no. 3, pp. 381–414, 2024.

[21] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.

[22] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[23] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[24] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[25] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[26] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[27] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundation of Robotics VIII: Selected Contributions of the Eight International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2009, pp. 449–464.

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

[29] S. Chitta, I. Sucan, and S. Cousins, "Moveit!" *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[30] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *arXiv preprint arXiv:1404.3785*, 2014.