Generating and Optimizing Topologically Distinct Guesses for Mobile Manipulator Path Planning

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

***Abstract*— Optimal path planning often suffers from getting stuck in a local optimum. This is often the case for mobile manipulators due to nonconvexities induced by obstacles and robot kinematics. This paper attempts to circumvent this issue by proposing a pipeline to obtain multiple distinct local optima. By evaluating and selecting the optimum among multiple distinct local optima, it is likely to obtain a closer approximation of the global optimum. We demonstrate this capability in optimal path planning of nonholonomic mobile manipulators in the presence of obstacles and subject to end effector path constraints. The nonholomicity, obstacles, and end effector path constraints often cause direct optimal path planning approaches to get stuck in local optima. We demonstrate that our pipeline is able to circumvent this issue and produce a final local optimum that is close to the global optimum.**

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, etc., cause the NLP to be highly non-convex. 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. To this end, we introduce the notion of "multi-locally optimal" solutions. We call a solution to an optimization problem *multi-locally optimal* if it is the optimal solution among multiple distinct local optima.

The challenge of obtaining a multi-locally optimal path is computing multiple distinct local optima since most research has only been 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*. Generating and optimizing these homotopically distinct initial guesses is the primary focus of this work.

Rufus Wong, Mayank Sewlia, Adrian Wiltz and Dimos V. Dimarogonas 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

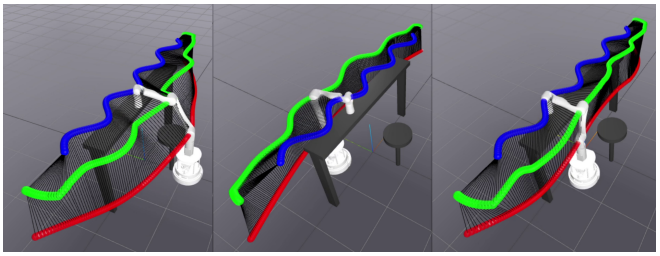


Fig. 1: Mobile manipulator executing three homotopically distinct locally optimal paths given a desired end effector path. Blue shows the desired end effector path, green and red shows computed the elbow and base paths respectively.

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. This form was chosen for its cost-effectiveness and versatility. We further require that the end effector follows a predetermined path. Such end-effector path constraints arise naturally in applications such as painting or wiping a table.

The contribution of this paper is the development of a path planning pipeline for nonholonomic mobile manipulators that handles 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 modify NAGS in order to leverage the specific structure of the configuration graph for improved 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 each of the stages 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 discusses our approach and draws a conclusion.

II. Related Work

A. Mobile Manipulators

Path planning for mobile manipulators has been widely studied. The survey [5] provides a comprehensive overview on mobile manipulators planning algorithms. We highlight a

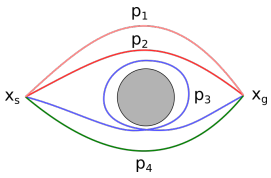


Fig. 2: Given the grey obstacle, p_1 and p_2 belong to the same $\mathcal{H}$-class (homotopically equivalent) while p_2, p_3, p_4 all belong to different $\mathcal{H}$-classes (homotopically distinct).

few results in the context of planning with end effector path constraints on nonholonomic mobile manipulators.

A rapidly exploring random tree (RRT) [6] based approach is introduced in [7] that handles constraints via tangent space projection. This has been further generalized and incorporated into OMPL [8]. Such sampling approaches can often be postprocessed to produce a locally optimal path. A genetic algorithm is proposed in [9] which is probabilistically optimal but may take excessive amounts of time to achieve global optimality. An inverse kinematics (IK) based method is proposed in [10] which produces a locally optimal solution greedily. None of these approaches provide a guaranteed way of discovering more than one distinct local optimum.

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 a nonconvex cost function only leads to locally optimal results.

C. Topological Path planning

This 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. 2). Many probabilistic methods for finding homotopically distinct paths have been proposed [12]–[15]. However, they generally scale poorly to high dimensions. As such, using a lower dimensional or simpler topological path planning setup for high-level 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 [16], [17], quadrotors [18] and manipulators [19], [20].

For applying this topological and optimal path planning pipeline to mobile manipulators, a major problem is the determination of $\mathcal{H}$-class, which is a highly non-trivial task that becomes increasingly complex for high-dimensional configuration spaces. In [4], a novel Neighborhood Augmented Graph Search (NAGS) algorithm has been recently 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 for a simpler, lower-dimensional problem and use optimal path planning for the local refinement of the full-dimensional original problem.

III. Problem Formulation

The path planning problem concerns a 6-degree-of-freedom(DoF) elbow manipulator attached to a nonholonomic differential drive mobile base. 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

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

where $v \in \mathbb{R}$ is the linear and $\omega \in \mathbb{R}$ is the angular velocity.

For ease of notation, define $x_{b\perp} = [x_b, 0]^T = [x, y, 0]^T$. Given the base position x_b, the arm is 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

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

This describes the arm kinematics. In particular, 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

$$\dot{x}_w = c, \quad \dot{x}_e = g \tag{3}$$

where $c \in \mathbb{R}^3$ is the elbow and $g \in \mathbb{R}^3$ is the end effector velocity.

Given the base and arm kinematics and dynamics above, the robot configuration q can be fully defined by

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

subject to the aforementioned constraints.

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 $x_e(t)$ with $t \in [0, 1]$ being the path parameter representing an auxiliary time normed to the interval $[0, 1]$ is given. The planning problem, then is to find a feasible path, satisfying kinematic and dynamic constraints, that does not collide with obstacles and minimizes

$$\int_0^1 v(t)^2 + \omega(t)^2 + c(t)^2 dt \tag{4}$$

The final solution will be of the form $Q = \{q(t_0), q(t_1), \ldots, q(t_T)\}$, $t_0 = 0$, $t_T = 1$ with t_i equally spaced and T being the number of path samples.

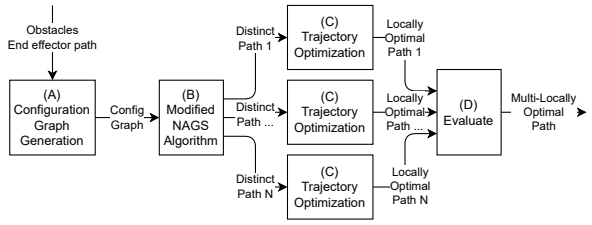


Fig. 3: The planning pipeline

Algorithm 1 Pipeline for finding multi-locally optimal paths

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

1: **function** FINDPATH($@x_e, n, dt, T$)
2: $G = (V, E) :=$ ConfigurationGraphGeneration(x_e)
3: $\begin{bmatrix} (x^\circ_{b1}, x^\circ_{w1}, t^\circ_1) \\ (x^\circ_{b2}, x^\circ_{w2}, t^\circ_2) \\ \vdots \\ (x^\circ_{bn}, x^\circ_{wn}, t^\circ_n) \end{bmatrix} :=$ modifiedNAGS(G, n)
4: **for all** $i \in 1 \dots n$ **do** ▷ can be run in parallel
5: $(\text{cost}_i, Q^\star_i) :=$ TrajOpt($x^\circ_{bi}, x^\circ_{wi}, t^\circ_i, dt, T$)
6: $i^\star := \arg\min_i(\text{cost}_i)$
7: **return** $Q^\star_{i^\star}$

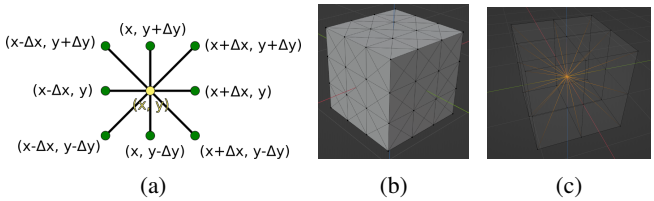
IV. Methodology

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). Each vertex of the graph represents a collision-free robot configuration, and each edge represents a collision-free transition between configurations.
(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 converted into 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 high-dimensional continuous space of collision-free robot configurations into a low-dimensional discrete graph for the subsequent NAGS algorithm. We reduce dimensionality as follows: Firstly, the base heading and nonholonomic constraints are ignored at this stage. Secondly, the dimensionality is further reduced 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 $[x_b, t, w]^T \in \mathbb{R}^2 \times [0,1] \times \{-1, 1\}$ by incorporating the end effector path constraint. To see this, notice that x_e is fully defined by the path parameter t. Furthermore, given x_b and x_e, there only exists two feasible elbow positions: elbow up and elbow down, represented by $w = 1$ and $w = -1$ respectively, with $w \in \{-1, 1\}$. 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 $[x, y, t, w]^T$ and $[x_b, x_w, x_e]^T$ interchangeably with the understanding that the former can always be mapped to the latter via common IK procedures [21].



(a) (b) (c)

Fig. 4: Illustration of King's graphs. (a) Connectivity of a vertex in a 2D King's graph. (b) Exterior view of a 3D King's graph. (c) Connectivity of a vertex in a 3D King's graph.

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 $(x, y, t, w) \in \mathbb{R}^2 \times [0,1] \times \{-1, 1\}$ by predefined discretization intervals $\Delta x, \Delta y, \Delta t$. This discretization determines the resolution of the graph and should be chosen based on the size of the smallest obstacle. The set of base positions $(x, y) \in \mathbb{R}^2$ which were originally unbounded, is also replaced by a bounded $(x, y) \in [-x_{\max}, x_{\max}] \times [-y_{\max}, y_{\max}]$ for some $x_{\max}$ and $y_{\max}$, defining the bounds of the base position. Then, we define the discretized bounded configuration space as

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

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 $(l_1 + l_2)$. As such, we define V as

$$\begin{aligned} V = \{(x, y, t, w) \in \mathfrak{C} : &\|[x, y, 0]^T - x_e(t)\|_2 \le l_1 + l_2 \\ &\wedge \neg \text{obs}(x, y, t, w)\} \end{aligned}$$

Notice that for each t, $[x, y, 0]$ describe a *disk* centered at $x_e(t)$ with radius $l_1 + l_2$.

Edges between vertices represent possible transitions between the robot configurations. An edge $e = (v_1, v_2)$ between a pair of vertices $v_1 = (x_1, y_1, t_1, w_1)$ and $v_2 = (x_2, y_2, t_2, w_2)$ is in E if either of the following hold

1) $\text{kgc}(v_1, v_2) \wedge \neg\overline{\text{obs}}(v_1, v_2) \wedge (w_2 = w_1)$
2) $\text{kgc}(v_1, v_2) \wedge \neg\overline{\text{obs}}(v_1, v_2)$
$\wedge (w_2 \neq w_1) \wedge \|[x_1, y_1, 0]^T - x_e(t)\|_2 = l_1 + l_2$

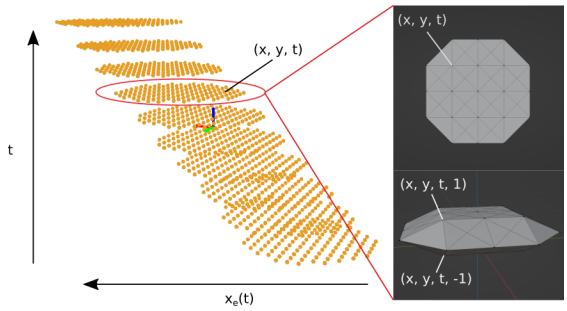


Fig. 5: An example of a CG with end effector path defined by $x_e(t) = [t, 0, 0]^T$

where kgc, the King's Graph Condition, is defined as

$$\text{kgc}(v_1, v_2) = \text{True} \iff$$
$$\exists \alpha \in \{-1, 0, 1\}, \exists \beta \in \{-1, 0, 1\}, \exists \gamma \in \{-1, 0, 1\}$$
$$: (x_2 = x_1 + \alpha \Delta x) \wedge (y_2 = y_1 + \beta \Delta y) \wedge (t_2 = t_1 + \gamma \Delta t)$$

The King's Graph Condition expresses the connectivity of a 3D King's graph with axis x, y, t, analogous to the 2D King's graph in Fig. 4a. The function $\overline{\text{obs}}(v_1, v_2)$ checks whether any robot pose between v_1 and v_2 is in a collision, formally

$$\overline{\text{obs}}(v_1, v_2) = \text{True} \iff$$
$$\exists \alpha \in [0, 1] : \text{obs}(\alpha v_1 + (1 - \alpha) v_2) = \text{True}$$

In practice, this is checked for a finite discretization of α.

The condition 1) expresses that, within the same elbow configuration, vertices are connected in the manner of a 3D King's Graph. The condition 2) expresses that connections across elbow configurations only occur at the joint singularity, when $\|[x_1, y_1, 0]^T - x_e(t)\|_2 = l_1 + l_2$. This can be visualized as each *disk* having two separate sides, the top side corresponding to $w = 1$ and the bottom side corresponding to $w = -1$. These two sides are separate except at the boundaries where they meet.

An example CG is illustrated in Fig. 5 with an end effector path constraint of the form $x_e(t) = [t, 0, 0]^T$. As t increases, $x_e(t)$ moves in the $+x$ direction which creates disks that move alongside in the $+x$ direction.

The edge cost of the CG represents the cost of transitioning from one robot configuration to another. The edge cost $d(v_1, v_2)$ between two vertices $v_1 = (x_1, y_1, t_1, w_1)$ and $v_2 = (x_2, y_2, t_2, w_2)$ is defined as the Euclidean distance along the (x, y, t) directions as follows:

$$d(v_1, v_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (t_2 - t_1)^2}$$

The cost on end effector motion t is to avoid back and forth motions along the end effector path incurring no cost.

B. Modified Neighborhood Augmented Graph Search

Our approach is based on NAGS [4, Algorithm 1]. The main idea behind it is to add the notion of homotopic equivalence to Dijkstra's Algorithm [22]. This is done by using a vertex's path neighborhood set (PNS), which is computed by running a reverse A* search [23] on the graph

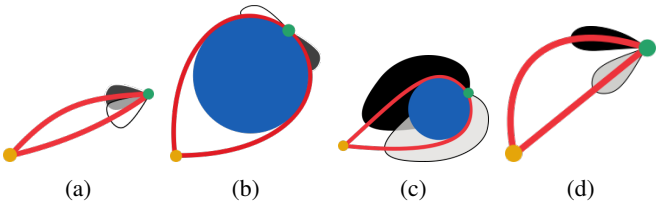


Fig. 6: Two paths starting from the yellow vertex and ending at the green vertex, their PNSes marked black and white, along with a blue obstacle. (a) Homotopically equivalent paths have overlapping PNS. (b) Homotopically distinct geodesic paths have non-overlapping PNS. (c) Having r too large causes false negatives. (d) Having r too small causes false positives.

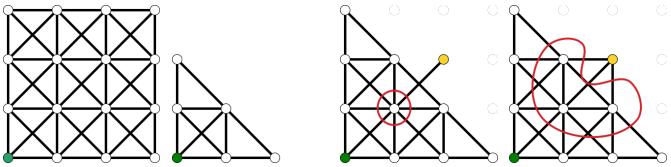


Fig. 7: (a) shows the underlying CG. (b)-(d) shows various iterations of NAGS. Green is the starting vertex. Yellow is the vertex ζ. $\mathcal{P}(\zeta)$ is circled in red.

for a fixed search depth r from the vertex in question back to the starting vertex. The key to the algorithm is that the PNS is an approximation of the path tangent and can be used to differentiate between homotopically distinct paths. This is illustrated in Fig. 6a and 6b. We refer the reader to [4, Algorithm 1 and 3] for further details.

The usage of PNS has the problem that the search depth r has to be fine-tuned depending on the underlying graph structure and obstacle size to ensure that it correctly identifies homotopically distinct paths. Fig. 6c and Fig. 6d show potential situations where false negatives and false positives may occur. This is because the PNS is only calculated once per vertex and does not get updated as the connectivity of that vertex changes.

This fine-tuning is required because the original NAGS algorithm makes no assumption on the graph structure. However, due to the King's-graph-like structure of our configuration graph generated in the previous section, we can specialize the NAGS algorithm to exploit this structure for improved accuracy.

Our modifications are based on the following observation regarding the NAGS Algorithm on a King's Graph: Given a NAG vertex ζ and the current state of the NAG $G_N = (V_N, E_N)$, define the parent set (PS) of ζ as follows

$$\mathcal{P}(\zeta) := \{\psi : \exists (\zeta, \psi) \in E_N\}$$

Observe that as NAGS updates G_N in the absence of obstacles, the first vertex ψ added to $\mathcal{P}(\zeta)$ must be on the shortest path to ζ. Any subsequent vertex added to $\mathcal{P}(\zeta)$ must be adjacent to an existing vertex in $\mathcal{P}(\zeta)$. This is illustrated in Fig. 7.

Thereby, we motivate the use of PS as an alternative to

Algorithm 2 Parent Set (PS) Computation

Require:
ζ: Vertex to compute PS for
G_N: Current NAG $G_N = (V_N, E_N)$
function COMPUTEPS(ζ, G_N)
  $\mathcal{P} := \{\psi : \exists(\zeta, \psi) \in E_N\}$
  return $\mathcal{P}$

PNS, replacing [4, Algorithm 3] with Algorithm 2. The PS of a vertex is updated whenever an edge is added to the vertex. We further update the equivalence relation ($\equiv$) between vertices of the NAG $G_N = (V_N, E_N)$

$$\zeta_1 \equiv \zeta_2 \iff \zeta_1.cg = \zeta_2.cg \\ \wedge \exists \psi_1 \in \mathcal{P}(\zeta_1), \psi_2 \in \mathcal{P}(\zeta_2) : (\psi_1, \psi_2) \in E_N$$

where $\zeta_i.cg$ retrieves the CG vertex represented by NAG vertex ζ_i. This says that two vertices in a NAG are equivalent if they correspond to the same CG vertex and if their PS are adjacent.

The PS improves upon the PNS in two key ways. Firstly, it is equivalent to generating the PNS with $r = 1$ meaning it is equal to the smallest possible PNS. This avoids the problem illustrated in Fig. 6c. Secondly, the PS continuously expands as new homotopically equivalent vertices are discovered. This expansion allows subsequent homotopically equivalent vertices to have adjacent PSes, mitigating the problem of false positives in Fig. 6d.

With the modified NAGS algorithm, we can more effectively find a pre-specified number of homotopically distinct paths. These can then be used as initial guesses for the subsequent trajectory optimization.

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

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

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

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

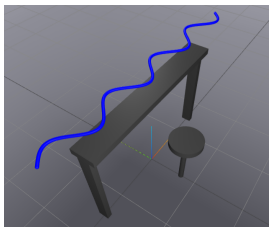
$$x_w[k] - x_b[k] = a(x_e[k] - x_b[k]) + \begin{bmatrix} 0 \\ 0 \\ b \end{bmatrix}, \tag{5d}$$

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

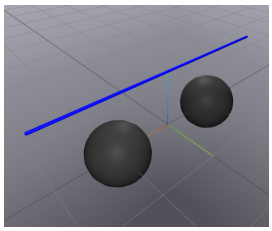
$$\theta[k+1] = \theta[k] + \omega[k]dt, \tag{5f}$$

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

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



(a) Planning Problem 1



(b) Planning Problem 2

Fig. 8: The planning problems

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

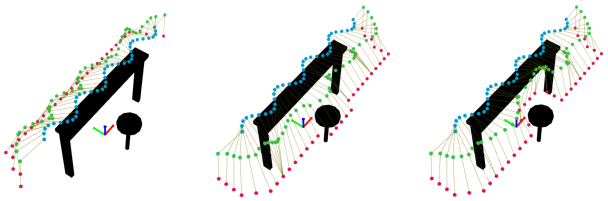
In this Section, we validate our pipeline in two ways:

- We show that our pipeline[1] can generate multi-locally optimal paths.
- We show that our CG generation and modified NAGS algorithm outperforms OMPL's constrained motion planning [8] in generating homotopically distinct initial guesses.

We consider the planning of a mobile manipulator in the form of a Kinova Gen3 robot arm attached to a Turtlebot 4, tasked with cleaning a counter table with a sine wave motion while avoiding collisions with the table itself and the bar chair nearby (Fig. 8a). For this problem, the aim is to find the multi-locally optimal path among three distinct local optima.

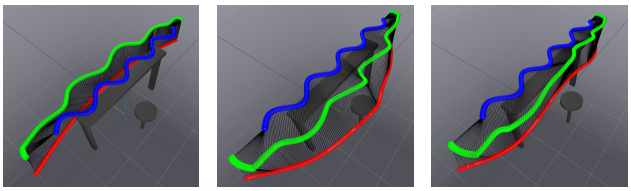
We generate the CG by discretizing the end effector path at 0.05m (5cm) intervals. The base position is discretized at a resolution of 0.1m. Edges in the CG are subsampled at 0.01m for collision checking. The final CG took 0.895s to generate. NAGS took 2.353s to generate three homotopically distinct paths, each belonging to a different $\mathcal{H}$-class, arbitrarily labelled with 1-3, shown in Fig. 9. For the NLP, we chose $T = 200$ and $dt = 0.2$. Without an initial guess, the optimizer failed to solve the NLP. Using the distinct paths as initial guesses, the optimizer was able to generate 3 distinct local optima, shown in Fig. 10. A snapshot of the mobile manipulator executing the path is given in Fig. 1. The NLP

[1] https://github.com/rcywongaa/topologically_distinct_guesses



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

Fig. 9: Results of NAGS



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

Fig. 10: Results of optimizing NAGS guesses

took, on average, 88.7s to solve. The path in $\mathcal{H}$-class 1 had the lowest cost, making it the multi-locally optimal path.

A. Comparison with OMPL

We now offer a brief comparison between our CG and modified NAGS algorithm and OMPL's constraint planning implementation [8] with respect to their ability to generate homotopically distinct initial guesses. We evaluate this based on three criteria:

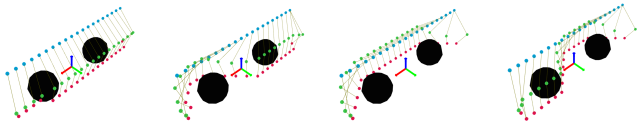
- Time required to generate the pre-specified number of homotopically distinct paths
- Variance in time required to generate the paths
- Validity of the generated paths as initial guesses where a path is a valid initial guess only if it allows the NLP to be solved successfully

The OMPL planner and planning scenario is set up using MoveIt [24], [25]. We use the KPIECE planner [26], an RRT-based planner, for our constraint planning problem. This choice is motivated by [8], which shows the KPIECE planner has superior performance compared to other planners when planning in high dimensional configuration space with end effector constraints.

For the table cleaning scenario and with a path tolerance of 0.05m (5cm), OMPL could not generate any path within a timeout of 60s. Hence we compare a simpler scenario illustrated in Fig. 8b. In this scenario, our pipeline and OMPL are both tasked with finding four homotopically distinct paths.

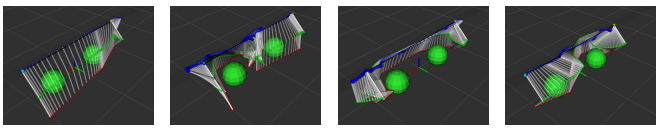
For our pipeline, we generate the CG by discretizing the end effector path at 0.1m (10cm) intervals. Base positions are also discretized at a resolution of 0.1m. Edges in the CG are further subsampled at 0.01m for collision checking. The final graph took 0.141s to generate. The average runtime for NAGS is 0.496s. This generated four paths, each belonging to a different $\mathcal{H}$-class, arbitrarily labelled with 1-4 (see Fig. 11).

For OMPL, we set a path tolerance of 0.05m, assuming the worst-case violation for NAGS is half the path discretization



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

Fig. 11: Results of NAGS for the simple planning problem belonging to different $\mathcal{H}$-classes



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

Fig. 12: Results of OMPL constraint planning belonging to different $\mathcal{H}$-classes

interval. OMPL, on average, requires 17 attempts to generate a path for each of the four $\mathcal{H}$-classes. Example results from each $\mathcal{H}$-class are shown in Fig. 12. A detailed comparison is provided in Table I. We see that our CG + modified NAGS pipeline outperforms OMPL in terms of runtime, variance in runtime and validity of generated paths.

VI. Discussion & Conclusion

This paper presents a pipeline for nonholonomic mobile manipulator path planning in the presence of end effector constraints that achieve multi-local optimality. This is achieved by generating and optimizing homotopically distinct guesses and finally choosing the best among the local optima. Experiments showed that our pipeline was able to generate *multi-locally optimal* solutions for a fairly complex table-cleaning scenario. Furthermore, the experiments showed our scheme for generating homotopically distinct paths outperformed the state-of-the-art motion planning library OMPL.

We note that, as with other graph-search-based algorithms, our algorithm also suffers from the curse of dimensionality, which would arise when applied to mobile manipulators with higher DoFs. Also, since each obstacle produces at least two homotopically distinct paths, the number of paths grows exponentially with the number of obstacles, thus leading to poor performance of the NAGS algorithm. As such, our pipeline generally excels in scenarios where constraints can reduce the dimensionality of the problem and where there are a small number of large obstacles leading to a smaller configuration graph. We argue that 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.

Algorithm	Avg. (stdev) runtime	% valid guesses
CG + Modified NAGS	0.637 (0.011)	100
OMPL	29.9 (22.9)	58.3

TABLE I: Results of our modified NAGS algorithm and OMPL constrained planning for the simple planning problem

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 3d," 2023.

[5] T. Sandakalum and M. H. Ang Jr, "Motion planning for mobile manipulators—a systematic review," *Machines*, vol. 10, no. 2, p. 97, 2022.

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

[7] 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.

[8] 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.

[9] J. Vannoy and J. Xiao, "Real-time adaptive motion planning (ramp) of mobile manipulators in dynamic environments with unforeseen changes," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1199–1212, 2008.

[10] F. G. Pin, C. J. Hacker, K. B. Gower, and K. A. Morgansen, "Including a non-holonomic constraint in the fsp (full space parameterization) method for mobile manipulators' motion planning," in *Proceedings of International Conference on Robotics and Automation*, vol. 4. IEEE, 1997, pp. 2914–2919.

[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] K. Kolur, S. Chintalapudi, B. Boots, and M. Mukadam, "Online motion planning over multiple homotopy classes with gaussian process inference," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2358–2364.

[14] D. Yi, M. A. Goodrich, and K. D. Seppi, "Homotopy-aware rrt*: Toward human-robot topological path-planning," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2016, pp. 279–286.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] C. D. Toth, J. O'Rourke, and J. E. Goodman, *Handbook of discrete and computational geometry*. CRC press, 2017.

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

[25] 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.

[26] 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.

[27] M. Farber, "Bridged graphs and geodesic convexity," *Discrete mathematics*, vol. 66, no. 3, pp. 249–257, 1987.

APPENDIX

A. Modified NAGS Algorithm

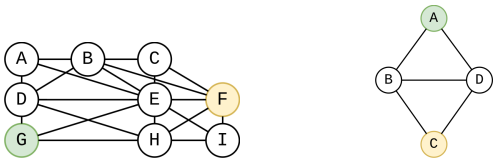
Algorithm 3 Modified NAGS Algorithm

Require:
$q_s \in V$: Start configuration
$\mathcal{N}_G$: Neighbor/successor function for graph G
$\mathcal{C}_G : V \times V \to \mathbb{R}^+$: Cost function
@stopSearch: $V_N \to \{0, 1\}$: Stopping criteria
@computePS: $V \times (V_N, E_N)$: parent set computation

Ensure: G_N: Graph with costs and parent set for every vertex

function SEARCHNAG(q_s, $\mathcal{N}_G$, $\mathcal{C}_G$, @stopSearch)
  $v_s := (q_s, \{q_s\})$ ▷ start vertex in V_N, with self-reference in its path neighborhood set
  $g(v_s) := 0$ ▷ path distances
  $V_N := \{v_s\}$ ▷ vertex set
  $E_N := \emptyset$ ▷ edge set
  $Q := \{v_s\}$ ▷ open list (heap data structure)
  $v := v_s$
  while $Q \neq \emptyset \wedge \neg$stopSearch$(v)$ **do**
    $v := (q, U) = \arg\min_{v' \in Q} g(v')$ ▷ heap pop
    $Q = Q - v$ ▷ heap pop
    $U' =$ computePS$(v, (V_N, E_N))$
    for all $q' \in \mathcal{N}_G(q)$ **do**
      $v' := (q', U')$ ▷ successor
      $g' = g(v) + C_G(q, q')$ ▷ distance for v'
      if $\nexists w \in V_N$, with $v' \equiv w$ **then** ▷ new vertex
        $V_N = V_N \cup \{v'\}$
        $E_N = E_N \cup \{(v, v')\}$
        $g(v') = g'$
        $Q = Q \cup \{v'\}$
        v'.came_from $= v$
      else ▷ vertex already exists (w)
        $w = v'$
        $E_N = E_N \cup \{(v, w)\}$
        if $g' < g(w) \wedge w \in Q$ **then** ▷ update w
          $g(w) = g'$
          w.came_from $= v$
          $w.U = U'$
  return $G_N = (V_N, E_N)$

The modifications to [4, Algorithm 1] are presented in Algorithm 3, with changes highlighted in blue. The PNS is



(a) Non-uniformly discretized CG with $GH < GD + DB < GE$ (b) Simple bridged graph with $AB = BC = AD = DC = BD$

Fig. 13: Different graph structures

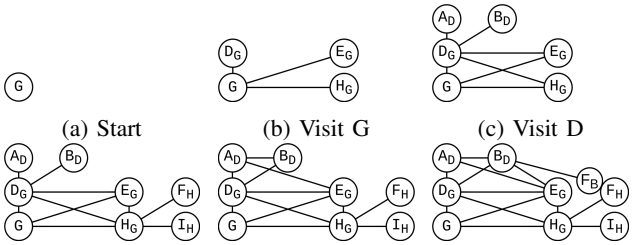


Fig. 14: Progression of the modified NAGS algorithm. Subscript indicates the parent of the vertex.

replaced with the PS in line 11.

B. Generalizing the modified NAGS algorithm to different graph structures

With a few more modifications (highlighted in orange and purple), we may generalize the modified NAGS algorithm to more flexible graph structures illustrated in Fig. 13. This is the version implemented for this paper.

1) Non-uniform discretization: Consider a non-uniformly discretized CG in Fig. 13a where we wish to find homotopically distinct paths from vertex G to vertex F. Note that there is only 1 homotopically unique path between G and F. The modified NAGS algorithm proceeds as shown in Fig. 14. Consider the last step in Fig. 14f. $\mathcal{P}(F_B) = \{B_D\}$, $\mathcal{P}(F_H) = \{H_G\}$. Since $\nexists \psi_1 \in \mathcal{P}(F_B), \psi_2 \in \mathcal{P}(F_H) : (\psi_1, \psi_2) \in E_N$, we have $F_B \not\equiv F_H$. Hence the algorithm incorrectly determines that there are two homotopically distinct paths from G to F. This is due to the fact that the modified NAGS algorithm (as well as the original NAGS algorithm), adds a vertex to the NAG based on the parent of that vertex. This motivates our changes in line 25-26 and line 11-12. This ensures that vertices are added to the NAG in the order of the cost to the vertex itself, rather than the parent.

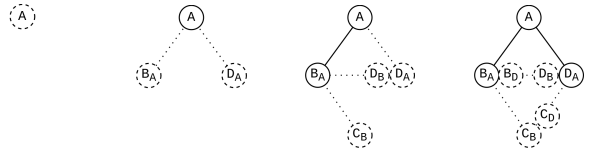


Fig. 15: Progression of the modified NAGS algorithm. Subscript indicates the parent of the vertex. Dotted vertices represent vertices in the heap.

Algorithm 4 Generalized Modified NAGS Algorithm

Require:
$q_s \in V$: Start configuration
$\mathcal{N}_G$: Neighbor/successor function for graph G
$\mathcal{C}_G : V \times V \to \mathbb{R}^+$: Cost function
@stopSearch: $V_N \to \{0, 1\}$: Stopping criteria
@computePS: $V \times (V_N, E_N)$: parent set computation
Ensure: G_N: Graph with costs and parent set for every vertex

1: **function** SEARCHNAG(q_s, $\mathcal{N}_G$, $\mathcal{C}_G$, @stopSearch)
2: $v_s := (q_s, \{q_s\})$
3: $g(v_s) := 0$
4: $V_N := \{v_s\}$
5: $E_N := \emptyset$
6: $Q := \{v_s\}$
7: $v := v_s$
8: **while** $Q \neq \emptyset \wedge \neg$stopSearch$(v)$ **do**
9: $v := (q, U) = \arg\min_{v' \in Q} g(v')$
10: $Q = Q - v$
11: $V_N = V_N \cup \{v\}$ ▷ add vertex when visiting
12: $E_N = E_N \cup \{(v, v.\text{came\_from})\}$
13: $U' =$ computePS$(v, (V_N, E_N))$
14: **for all** $q' \in \mathcal{N}_G(q) : \exists w \in \mathcal{N}_G(q) \wedge (q', U') \equiv w$ **do** ▷ handle equivalent vertices first
15: $v' := (q', U')$
16: $g' = g(v) + C_G(q, q')$
17: $w = v'$
18: $E_N = E_N \cup \{(v, w)\}$
19: **if** $g' < g(w) \wedge w \in Q$ **then**
20: $g(w) = g'$
21: $w.\text{came\_from} = v$
22: $w.U = U'$
23: **for all** remaining $q' \in \mathcal{N}_G(q)$ **do**
24: $v' := (q', U')$ ▷ guaranteed new vertices
25: ~~$V_N = V_N \cup \{v'\}$~~ ▷ do not add vertex here
26: ~~$E_N = E_N \cup \{(v, v')\}$~~
27: $g(v') = g'$
28: $v'.\text{came\_from} = v$
29: $Q = Q \cup \{v'\}$
30: **return** $G_N = (V_N, E_N)$

2) Triangulated/bridged graph structure [27]: Consider the bridged CG in Fig. 13b with progression shown in Fig. 15. Note that there is only 1 homotopically unique path between A and C. Consider the final step in Fig. 15d. 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 $C_B \not\equiv C_D$. To tackle this nondeterministic behavior, we prioritize processing equivalent vertices first (line 14) before processing nonequivalent vertices (line 23).