IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

SMUG Planner: A Safe Multi-Goal Planner for Mobile Robots in Challenging Environments

Changan Chen[1], Jonas Frey[1,2], Philip Arm[1], and Marco Hutter[1]

***Abstract*—Robotic exploration or monitoring missions require mobile robots to autonomously and safely navigate between multiple target locations in potentially challenging environments. Currently, this type of multi-goal mission often relies on humans designing a set of actions for the robot to follow in the form of a path or waypoints. In this work, we consider the multi-goal problem of visiting a set of pre-defined targets, each of which could be visited from multiple potential locations. To increase autonomy in these missions, we propose a safe multi-goal (SMUG) planner that generates an optimal motion path to visit those targets. To increase safety and efficiency, we propose a hierarchical state validity checking scheme, which leverages robot-specific traversability learned in simulation. We use LazyPRM* with an informed sampler to accelerate collision-free path generation. Our iterative dynamic programming algorithm enables the planner to generate a path visiting more than ten targets within seconds. Moreover, the proposed hierarchical state validity checking scheme reduces the planning time by 30% compared to pure volumetric collision checking and increases safety by avoiding high-risk regions. We deploy the SMUG planner on the quadruped robot ANYmal and show its capability to guide the robot in multi-goal missions fully autonomously on rough terrain.**

***Index Terms*—Motion and Path Planning; Constrained Motion Planning**

I. INTRODUCTION

MULTI-GOAL missions are commonly seen in exploration, inspection, and monitoring scenarios. In those missions, a robot needs to visit numerous targets of interest for detailed investigation, for example to deploy instruments, collect samples, or read measurements from gauges. However, in current missions, the autonomy of the robot to navigate and plan is limited and relies on human operators [1]–[3].

Space exploration is a common application of such multi-goal missions. One example is the geological mission I of the ARCHES analog demonstration mission [1]. In this mission, a flying system and a rover map the environment and detect multiple targets of interest. A second rover collects samples and returns them to the landing system one at a time according to scientists' prioritization. A planner that generates a global path visiting all identified targets according to the map provided by other robots could increase the autonomy of the robot team. Another space exploration example is the Space Resources Challenge (SRC) launched by the European Space Agency (ESA) and the European Space Resources Innovation Centre (ESRIC) [2] that simulates a lunar prospecting mission, in which multiple targets need to be mapped and characterized. Several teams used a multi-robot strategy to map the environment and find the targets before starting a close-up investigation of the targets. However, none of the teams used a multi-goal planner: They either operated via waypoints provided by an operator or used heuristics to always visit the next closest target.

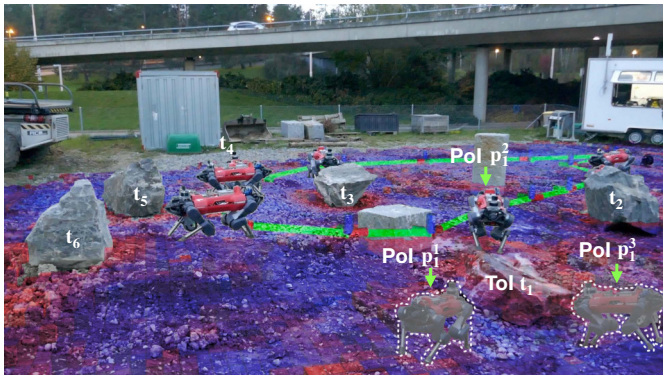


Fig. 1: The ANYmal quadruped robot autonomously performs a multi-goal mission of visiting six stones (Targets of Interest (ToIs):t_i) at one of the corresponding Pose of Interest (PoI): p_i^j following the global path colored in green. The traversability map is overlaid on the real scene. The red regions are untraversable whereas the blue regions are traversable. The PoIs p_1^1, p_1^2, p_1^3 for the ToI t_1 are indicated by the green arrows. p_1^2 is selected, whereas the other two not selected PoIs are gray.

Apart from exploration missions, multi-goal navigation is also relevant for industrial inspection and monitoring tasks. An example is the ARGOS (Autonomous Robot for Gas and Oil Sites) Challenge [4] initiated by the company Total and ANR, in which the robots are required to navigate over a decommissioned gas dehydration skid to perform inspection tasks, such as reading sensor dials and valve positions, at various targets. Gehring et al. [3] provide a solution to the industrial inspection of an offshore HVDC platform with the ANYmal quadruped robot. After the environment is mapped using the onboard sensors, the operator records the desired target location, at which detailed visual or thermal inspection needs to be done, and defines the via-point for the robot.

Manuscript received: March, 1, 2023; Revised May, 31, 2023; Accepted July, 23, 2023. This paper was recommended for publication by Editor Hanna Kurniawati upon evaluation of the Associate Editor and Reviewers' comments. This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication (NCCR dfab). This project has received funding through ESA contract nos. 4000137333/22/NL/AT and 4000135310/21/NL/PA/pt. This work has been conducted as part of ANYmal Research, a community to advance legged robotics. This work was supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 101016970, No 101070405. Jonas Frey is supported by the Max Planck ETH Center for Learning Systems.

[1]Authors are with ETH Zurich, Robotics Systems Lab; Leonhardstrasse 21, 8092 Zurich, Switzerland.

[2]The author is with the Max Planck Institute for Intelligent Systems

Code available at https://github.com/leggedrobotics/smug_planner

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

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

There is currently a lack of a planner that can efficiently generate an optimal path visiting multiple goals while taking safety into account, which is crucial for achieving fully autonomous and successful multi-goal mission execution. This results in such missions mostly relying on human operators, causing delays, sub-optimal decision-making, and limiting efficiency.

In this paper, we present SMUG planner, a multi-goal planner to plan a safe path to visit a set of targets at the poses in their vicinity for detailed investigation. We refer to the targets as Target of Interest (ToI) and the valid poses to visit them as Pose of Interest (PoI), as illustrated in Fig. 1. More precisely, we present a planner to visit one PoI per ToI, which we refer to as multi-goal mission in this paper. This is the Generalized Traveling Salesman Problem (GTSP) with collision-free path planning. The large number of collision-free paths involved in the problem makes it impossible to simply deploy an off-the-shelf GTSP solver, such as GLNS [5], which would require computing all collision-free paths and results in infeasible computational cost. How to efficiently ensure the safety and the optimality of the global path makes this an especially challenging problem.

SMUG planner implements a two-stage planning schema: At first the optimal visit sequence of the ToIs using a Traveling Salesman Problem (TSP) [6] solver is determined. Secondly, the optimal PoI to visit each ToI is determined using Dynamic Programming (DP) in an iterative fashion, which we refer to as Iterative Dynamic Programming (IDP). We use LazyPRM* [7] with an informed sampler to efficiently generate the optimal collision-free paths. To guarantee safety, we propose a hierarchical obstacle avoidance strategy using a robot-specific traversability estimation module [8]. This design allows for replanning during the mission and onboard deployment. SMUG planner can therefore guarantee autonomy in communication-denied environments, which are common in many exploration and monitoring scenarios.

Our contributions consist of the following:

- A multi-goal path planner for mobile robots solving the GTSP with collision-free paths that can generate a safe path to visit more than ten targets in seconds.
- A novel obstacle avoidance scheme that reduces the planning time by 30% compared to the pure volumetric collision checking and avoids high-risk regions.
- Hardware deployment on the ANYmal legged robot in a multi-goal mission on rough terrain. To the best of our knowledge, this is the first time a global planner is deployed to a real-world GTSP with collision-free path planning on a mobile robot.

II. RELATED WORK

Our work builds upon prior work developed in the context of sampling-based path planning and methods for solving the TSP and its variants modeling multi-goal problems. Given our objective of real-world deployment, we specifically review existing global planners deployed on mobile robots.

a) Sampling-Based Path Planning: One family of popular path planning methods is sampling-based path planners [9]–[12], which allows for a continuous state space instead of discretization as in grid-based methods. Probabilistic Roadmap (PRM) [13], and its optimal version PRM* [7] build a probabilistic roadmap for the environment and effectively reuse the information collected across different queries, making these planners suitable for planning across multiple start-goal pairs in the same environment. Because the collision-checking of nodes and edges is costly and efficiency is important to our planner, LazyPRM* [14] is well-suited, since it only checks edge validity if it may construct the optimal path, which reduces the planning time. Gammel et al. [15] proposed to restrict the sampling space based on the cost of the best path found so far, thus accelerating path improvement. Our method builds upon LazyPRM* with an informed sampler to achieve efficient path planning in multi-goal missions.

b) Multi-Goal Sequencing Problem: TSP is the problem of finding the optimal sequence visiting a set of targets. A TSP instance considers each target as a single point. Its two variants, GTSP [16] and TSP with Neighborhoods (TSPN) [17], consider the targets as sets of points and continuous neighborhoods respectively. This approach is more accurate in many real-world scenarios, where a point often only needs to be approached instead of visited exactly. Exact algorithms proposed to solve GTSP include using Branch and Cut [18], Lagrangian relaxation [19] or transforming GTSP to an equivalent TSP instance [20]. Other works solving GTSP and TSPN based on heuristics such as the ant-colony optimization [21], genetic algorithm [22], [23] and Lin-Kernighan heuristic [24] or formulate the problem as Mixed Integer Nonlinear Programming (MINLP) [25]. However, these works ignore the computationally expensive collision-free path planning for a mobile robot navigating in complex environments. Alatartsev et al. [26] divide the TSPN into two subproblems: Given the visiting point of each target, find the optimal visiting sequence. And given the visiting sequence, find the optimal visiting points. These subproblems are solved iteratively. Although the obstacle avoidance necessary in a real-world scenario is ignored, the strategy of splitting the TSPN into two subproblems is insightful, and we adopt it in our method. Gentilini [23] focuses on combinatorial optimization with Hybrid Random-Key Generic Algorithm (HRKGA) while assuming all path costs are known a priori. However, the author provides two cases in simulation that consider obstacle avoidance, which is achieved by planning the required collision-free path for each chromosome in the genetic algorithm, making it expensive and thus unsuitable for our problem that requires the fast online generation of safe paths.

The variants of GTSP and TSPN addressing collision-free path planning are called Multi-Goal Path Planning Problem (MTP) [27] and MTP for Goal Regions (MTPGR) [28] respectively. Gao et al. [29] divide the MTPGR into two subproblems as in [26]. However, instead of iterating between them, they first solve for the optimal sequence once. Then, the path between any two points is initially assumed to be the straight line, and refined by alternating iteratively using the Rubber Band Algorithm (RBA) [30] to find the visiting point and planning the needed path accordingly. Thus, collision-free paths are only planned if they may construct the optimal global path, largely reducing the number of paths to plan. However,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

the RBA is prone to locally optimal path cost, since it adjusts the visiting point solely based on the two neighboring paths. Moreover, the authors only illustrate their method on a 2D grid map and lack real-world demonstration.

Instead of solving MTPGR, we model the multi-goal problem as an MTP by assuming the ToIs to be discrete sets of PoIs rather than a continuous neighborhood, because one may want to deploy an instrument to the ToIs, which requires a certain angle of attack or impose other constraints, resulting in disconnected valid PoIs. However, the most general choice is to model each ToI as a set of disconnected neighborhoods, which results in GTSP with Neighborhoods (GTSPN) [31]. Our assumption of each ToI being a discrete set simplifies the problem, while still allowing for multiple valid PoI proposals.

c) Planners for Mobile Robots: Multiple existing planners for mobile robots have been successfully deployed in the real world [10], [11], [32], [33]. Although they are not designed for the multi-goal mission, we use previously proven concepts and design patterns.

Several works adopted a bifurcated global and local planning structure [11], [32], [33], where the local planner has detailed information in the vicinity of the robot, handles obstacle avoidance, and is guided by a global planner with coarser paths. Following this structure, we generate the entire traversing path and use it to guide a local planner that refines the path locally at a higher frequency.

A discrete volumetric map is often used to represent the environment [10], [11], [32], which can be obtained by Voxblox [34]. In [11] [10], and [32], the authors consider the volumetric collision with the environment, however, not accounting for terrain traversability, which is crucial for the feasibility and safety of the path followed by a local planner. Therefore, we propose to leverage an in simulation-trained legged robot-specific traversability module [8], within the global planner, to evaluate the robot-specific terrain traversability.

III. PROBLEM STATEMENT

We aim to solve the problem of finding the safe global path that visits every ToI in multi-goal missions. We formulate the problem as an MTP. The robot can visit a ToI at multiple PoIs in a neighborhood of it. In this work, we limit the problem to visiting one PoI per ToI.

Let $\mathcal{C}$ be the state space representing the environment and $\mathcal{C}_{collision}$ the set consisting of the states in $\mathcal{C}$ that result in a collision between the robot and environment. The collision-free space is therefore $\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{C}_{collision}$. A path is denoted as a discrete representation through a set of waypoints, i.e. $W(w_0, w_n) = \{w_0, w_1, w_2, \ldots, w_n\}$ with $w_i \in \mathcal{C}$. W represents the path starting at state w_0, traversing through w_1, w_2 ..., and ending at w_n. The cost of a path segment $\{w_i, w_{i+1}\}$ is given by a positive definite function $c(\cdot,\cdot) : \mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}_{\geq 0}$. The total cost of the path W is the sum of the cost of all its path segments, i.e.:

$$Cost(W(w_0, w_n)) = \sum_{i=0}^{n-1} c(w_i, w_{i+1}). \tag{1}$$

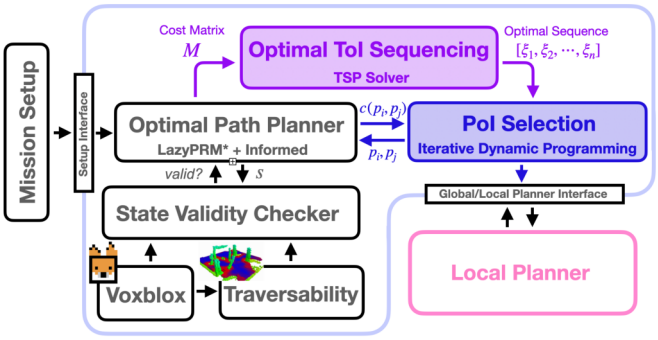


Fig. 2: System overview of the SMUG planner illustrating the information flow from accepting the mission to interacting with a low-level local planner. First, the optimal Target of Interest (ToI) sequence is computed (purple, Section IV-C). Second, the best Pose of Interest (PoI) is selected for each ToI (blue, Section IV-D). Our sampling-based optimal path planner (Section IV-A) is used in both steps to generate optimal collision-free paths. State validity is determined via geometric traversability checking in a volumetric map provided prior to the planning (Section IV-B).

The input and output of the problem are summarized as:

Input:

- Robot start and end pose $s \in \mathcal{C}_{free}$.
- A set of ToI poses $\mathcal{T} = \{t_i | t_i \in \mathcal{C}, i \in \{1, \cdots, n\}\}$ with $t_i \in \mathcal{C}$.
- For each ToI t_i, a set of n_i PoI $\mathcal{P}_i = \{p_i^j | p_i^j \in \mathcal{C}_{free}, j \in \{1, \cdots, m_i\}\}$.

Output: A collision-free path $W(w_0, w_n) = \{w_0, ..., w_n\}$ with minimum cost visiting every ToI $t_i \in \mathcal{T}$ at a PoI $p_i^j \in \mathcal{P}_i$, starting at s and ending at the same state:

$$\begin{aligned} \min_{W} \quad & Cost(W) \\ \text{s.t.} \quad & p_i^j \in W, \forall t_i \in \mathcal{T}, \exists p_i^j \in \mathcal{P}_i, \\ & w_0 = w_n = s, \\ & w_i \in \mathcal{C}_{free}, \forall w_i \in W \end{aligned} \tag{2}$$

The constraint $w_n = s$ can be relaxed but is used here to facilitate the notation. In this work, the state space $\mathcal{C}$ is chosen to be $SE(2)$, suitable for ground robots. A state $s$is thus expressed by its 2D coordinates x_s, y_s and yaw ψ_s as

$$s = \left[x_s, y_s, \psi_s\right]^\top. \tag{3}$$

For the cost function $c(\cdot,\cdot) : \mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}_{\geq 0}$, we use the Open Motion Planning Library (OMPL) [35] default distance function in $SE(2)$:

$$c(s_1, s_2) = w_t \left\| \begin{bmatrix} x_{s_1} \\ y_{s_1} \end{bmatrix} - \begin{bmatrix} x_{s_2} \\ y_{s_2} \end{bmatrix} \right\|_2^2 + w_r d(\psi_{s_1}, \psi_{s_2}), \tag{4}$$

where $w_t, w_r \in \mathbb{R}^+$ are the weights for the translational and the rotational cost, respectively, and $d(\cdot,\cdot) : [-\pi, \pi]^2 \mapsto [0, \pi]$ is defined as

$$d(\psi_1, \psi_2) = \min\left(|\psi_1 - \psi_2|, 2\pi - |\psi_1 - \psi_2|\right), \tag{5}$$

to compute the difference between two angles.

IV. METHOD

Solving the MTP directly is challenging due to the large number of possibly collision-free paths. Therefore, we adopt a two-step method similar to [29]. In the first step, we determine

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

the optimal sequence that visits every ToI by simplifying the problem to a TSP. This simplification is valid, assuming that the PoIs are close to the respective ToIs. In the second step, we select the PoI at each ToI to minimize the total path cost. Fig. 2 shows an overview of our system. The system consists of the following modules:

1) An optimal path planner generating the optimal path connecting two states (Section IV-A);
2) A state validity checker using the Truncated Signed Distance Field (TSDF) and the traversability map provided by Voxblox [34] and a traversability module [8], respectively, prior to the planning(Section IV-B);
3) A TSP solver computing the optimal visiting sequence of the ToIs given a cost matrix containing the path cost between each pair of ToIs (Section IV-C);
4) An IDP module selecting the optimal PoI for each ToI that minimizes the total path cost (Section IV-D);
5) A local planner [12] generating a finer path locally that follows the received global path.

A. Optimal Path Planning

We use LazyPRM* [14] with an informed sampler to generate the optimal collision-free paths between two states.

The sampler initially samples uniformly in $SE(2)$. After an initial path is found, it restricts the sample space based on the cost function and the current path cost to accelerate path optimization. Based on the informed sampler for the L_2 cost [15], we implement an informed sampler for the cost function $c(\cdot,\cdot)$ (Eq. (4)).

Assume the path to find starts at state s_1 and ends at state s_2, and an initial path with cost c_0 is already found. The new sample s_{new} are sampled in the ellipsoid

$$\mathcal{E}=\left\{\begin{bmatrix}x\\y\end{bmatrix}\middle|\left\|\begin{bmatrix}x\\y\end{bmatrix}-\begin{bmatrix}x_{s_1}\\y_{s_1}\end{bmatrix}\right\|_2^2+\left\|\begin{bmatrix}x\\y\end{bmatrix}-\begin{bmatrix}x_{s_2}\\y_{s_2}\end{bmatrix}\right\|_2^2<c_0-c_r\right\}, \quad (6)$$

where $c_r = d(\psi_{s_1}, \psi_{s_2})$. Any sample s outside this ellipsoid cannot improve the cost due to the following equation

$$\begin{aligned}
&c(s_1,s)+c(s,s_2)\\
=&\left\|\begin{bmatrix}x_{s_1}\\y_{s_1}\end{bmatrix}-\begin{bmatrix}x_s\\y_s\end{bmatrix}\right\|_2^2+\left\|\begin{bmatrix}x_s\\y_s\end{bmatrix}-\begin{bmatrix}x_{s_2}\\y_{s_2}\end{bmatrix}\right\|_2^2\\
&+d(\psi_{s_1},\psi_s)+d(\psi_s,\psi_{s_2}) \qquad (7)\\
\geq&\left\|\begin{bmatrix}x_{s_1}\\y_{s_1}\end{bmatrix}-\begin{bmatrix}x_s\\y_s\end{bmatrix}\right\|_2^2+\left\|\begin{bmatrix}x_s\\y_s\end{bmatrix}-\begin{bmatrix}x_{s_2}\\y_{s_2}\end{bmatrix}\right\|_2^2+d(\psi_{s_1},\psi_{s_2})\\
\geq&c_0-c_r+c_r=c_0.
\end{aligned}$$

Then, the yaw angle $\psi_{s_{new}}$ is sampled uniformly in $[-\pi, \pi]$. This procedure is summarized in Algorithm 1.

The sampled states are passed to a state validity checker to avoid states in collision with the environment.

B. State Validity Checker

We use a hierarchical obstacle avoidance scheme to ensure the feasibility and safety of the generated path efficiently. We adopt a robot-specific traversability module learned in simulation to accept the clearly safe states and discard clearly unsafe ones. Only states with unclear safety are checked for collision in an iterative volumetric fashion.

Algorithm 1: *Sample*()

```
if path found then
    c_r ← d(ψ_{s1}, ψ_{s2});
    c_0 ← best cost;
    x, y ← SampleInformedL2(s_1, s_2, c_0 − c_r);
else
    x ← SampleUniform(x_min, x_max);
    y ← SampleUniform(y_min, y_max);
end
ψ ← SampleUniform(−π, π);
s_new ← [x, y, ψ]^⊤;
return s_new
```

1) Traversability Filtering: We use a learned traversability map output by a convolutional neural network [8], which takes the occupancy map as its input. The network assigns for each voxel a traversability estimate ranging from 0 to 1 based on the success rate of traversing through this voxel in simulation as illustrated in Fig. 3a. We use this module to reduce the amount of iterative collision checking. The traversability is divided into three levels by setting two thresholds t_{low} and t_{high}, satisfying $0 \leq t_{low} \leq t_{high} \leq 1$. We obtained the traversability of a continuous state s defined in Eq. (3) by querying the corresponding traversability voxel. States with traversability less than t_{low} are directly invalidated, while those with traversability higher than t_{high} are validated. Only states with traversability between t_{low} and t_{high} are checked for collisions. This avoids checking collisions for obviously valid states, for instance, the states in the middle of a wide open space, as well as definitely invalid states that lie within an obstacle.

2) Volumetric Iterative Collision Checking: For the region with unclear traversability, we resort to the iterative volumetric checking for the bounding box of the robot base. The method uses the TSDF produced by Voxblox [34] to check if the box is in collision with the environment. To do so, the distance from the box center to the nearest environment surface is obtained from the TSDF map and compared to the box's outer and inner sphere radius. The box is in collision if the distance is smaller than the inner sphere radius, and not in collision if the distance is larger than the outer sphere radius. Otherwise, no conclusion is reached, and the algorithm divides the current box into two sub-boxes along its longest dimension and performs the same procedure on them until a predefined minimal sub-box resolution is reached and only the outer sphere is evaluated for collision (Fig. 4).

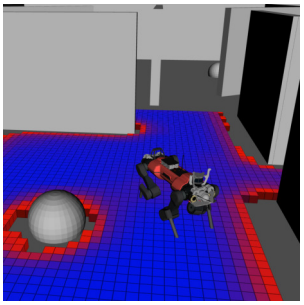
C. Optimal Sequencing (TSP)

To find the optimal visiting sequence of the ToIs, we first plan the collision-free paths connecting every pair of ToIs, resulting in the following cost matrix:

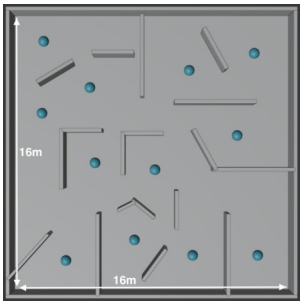
$$M=\begin{bmatrix}0 & m_{0,1} & m_{0,2} & \cdots & m_{0,n}\\ m_{1,0} & 0 & m_{1,2} & \cdots & m_{1,n}\\ \vdots & & \ddots & & \vdots\\ m_{n,0} & m_{n,1} & m_{n,2} & \cdots & 0\end{bmatrix}, \quad (8)$$

where $m_{i,j}$ is the cost of the collision-free path from t_i to t_j. For simplicity, we use $t_0 = s$. This matrix is passed to a

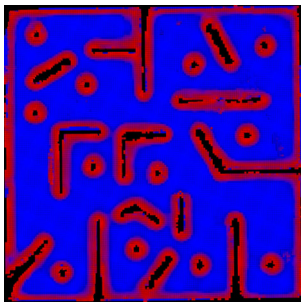
IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



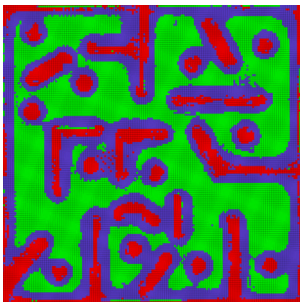
(a) Traversability estimation



(b) Sample environment

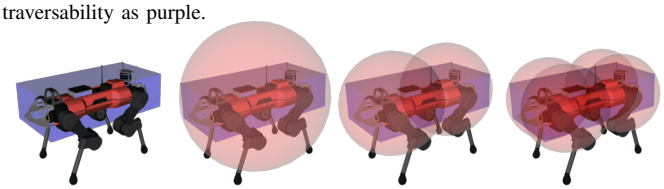


(c) Traversability map



(d) Traversability levels segmentation

Fig. 3: (a) The traversability network does not only estimate the traversability for voxels, but it also segments the obstacles out. No traversability voxels are placed around the obstacles, which prevents the planner from sampling here. (b) A sample indoor environment with twelve targets. (c) Traversability map of the sample environment produced with the traversability network, the traversability ranges from 0 to 1, colored from red to blue. (d) Segmentation of the traversability map into three levels using $t_{low} = 0.3, t_{high} = 0.8$. High traversability is marked as green, low traversability as red, and intermediate traversability as purple.



(a) Bounding box (b) 1st iteration (c) 2nd iteration (d) 3rd iteration

Fig. 4: The outer spheres (red) and the bounding box (blue) of the base of the ANYmal robot used in the first three iterations of the volumetric collision checking.

TSP solver [36] computing a near-optimal visiting sequence attempting to minimize:

$$\sum_{i=0}^{n-1} m_{\xi_i,\xi_{i+1}} + m_{\xi_n,\xi_0} \tag{9}$$

with $\xi_0 = 0$ and $[\xi_1, \cdots, \xi_n]^\top$ being a permutation of $[1, \cdots, n]$.

From now on we denote the i-th visited ToI as t_i, where $t_0 = s$. For each t_i, we denote the corresponding set of PoIs as $\mathcal{P}_i$. The collision-free path connecting two poses p_i, p_j is referred to as $W(p_i, p_j)$ in the following.

D. Iterative Dynamic Programming

After the visiting sequence is determined, the planner selects the PoI for each ToI. Given all collision-free paths between every PoI of two consecutively visited ToI, i.e. $W(p_{i,j}, p_{i+1,k}), \forall p_{i,j} \in \mathcal{P}_i, p_{i+1,k} \in \mathcal{P}_{i+1}$, the optimal path cost can be found via DP by solving the following equations

$$J_n^*(p_{n,j}) = Cost(W(p_{n,j}, s)), \tag{10}$$

$$J_i^*(p_{i,j}) = \min_{p \in \mathcal{P}_{i+1}} Cost(W(p_{i,j}, p)) + J_{i+1}^*(p), \quad \forall i \in \{1, \cdots, n-1\}, \tag{11}$$

$$J_0^*(s) = \min_{p \in \mathcal{P}_1} Cost(W(s, p)) + J_1^*(p). \tag{12}$$

The optimal PoIs to choose are thus obtained with

$$p_0^* = s, \tag{13}$$

$$p_i^* = \arg\min_{p \in \mathcal{P}_i} Cost(W(p_{i-1}^*, p)) + J_i^*(p), \quad \forall i \in \{1, \cdots, n\}. \tag{14}$$

DP is guaranteed to output the global optimum given all paths between PoIs of two consecutively visited ToIs. However, in the case of N ToIs and M PoIs per ToI, i.e. $|\mathcal{T}| = N, |\mathcal{P}_i| = M$, it requires planning $M^2(N-1) + 2M$ collision-free paths which increases quadratically with the number of PoIs. Eventually, not every path is used to form the optimal global path. Therefore, we propose to use DP in an iterative fashion and only plan the collision-free paths if they are needed to construct the optimal global path after each iteration. We define:

$$l(p_i, p_j) = \begin{cases} Cost(W(p_i, p_j)), & \text{if collision-free path} \\ & W(p_i, p_j) \text{ is planned,} \\ c(p_i, p_j), & \text{otherwise.} \end{cases} \tag{15}$$

This is the path cost of the collision-free $W(p_i, p_j)$ path between p_i and p_j, if it is already planned. Otherwise, $l(p_i, p_j)$ approximates the actual cost of $W(p_i, p_j)$ using the straight line between p_i and p_j, i.e. $c(p_i, p_j)$ defined in Eq. (4), which is a lower bound of $Cost(W(p_i, p_j))$.

At each iteration, we perform the introduced DP (Eqs. (10) to (14)), except replacing $Cost(W(\cdot, \cdot))$ by $l(\cdot, \cdot)$, to find the PoIs. Then we generate the collision-free paths connecting the two consecutively visited PoIs and update l accordingly. The algorithm terminates if the selected PoIs do not change over iterations. In this way, not necessarily all $M^2(N-1) + 2M$ collision-free paths need to be generated, reducing the total planning time. IDP preserves the optimality of DP if they use the same PoI to PoI paths, i.e.,$Cost(W(p_i, p_j))$, because the resulting path cost is lower than the lower bound of any other possible path.

Finally, we concatenate the path segments according to the obtained sequence of ToI and the selected PoI. We align the heading of the waypoints with the path direction (except for the ones at PoIs, which are fixed) to ensure that the robot moves forwards instead of sideways.

V. Experiments

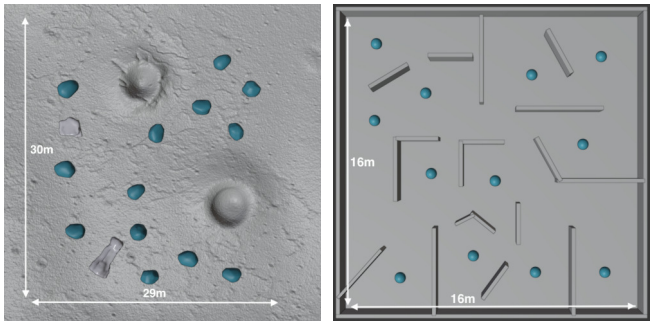
A. Simulation

We test our planner in two customized simulation environments, a lunar analog environment (Fig. 5a) and an indoor space (Fig. 5b), to illustrate its applicability in both exploration and industrial settings.

In the lunar environment, there are 13 ToIs, and ten PoIs are generated uniformly around each. In the indoor environment, we set 12 ToIs and for each two PoIs to choose from to

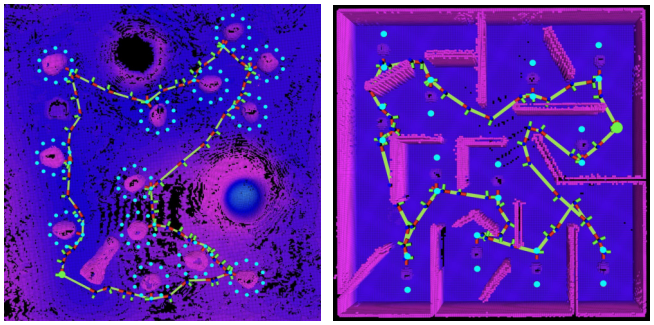
IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



(a) Lunar (b) Indoor

Fig. 5: (a) A lunar analog environment of $30m \times 29m$ with 13 stones to investigate (blue). (b) An indoor simulation environment of $16m \times 16m$ with 12 targets to inspect (blue)



(a) Lunar (b) Indoor

Fig. 6: Samples of generated paths for the lunar and the indoor environments, respectively. The feasible poses to choose are marked as cyan dots, and the robot start poses are marked as green dots.

represent smaller-size problems. Fig. 6 shows the generated path for the environments.

Since no other method exists that solves the presented problem, we compare the proposed method with the baseline designed ourselves. The following experiments are conducted on a laptop with Intel Core i7-9750H CPU and 32GB RAM. All means and standard deviations are calculated from ten consecutive runs.

1) IDP / DP / Iterative Rubber Band Algorithm (IRBA) Comparison: We compare the performance of the IDP with the method of preplanning all $M^2(N-1)+2M$ collision-free paths before performing DP once, which we refer to as DP, and an adaption of the method proposed in [29] used to solve MTPGR, which we refer to as Iterative RBA (IRBA).

Since IDP and IRBA do not necessarily require planning all $M^2(N-1)+2M$ paths, the planning time is largely reduced, as shown in Table I. In the lunar environment with 13 ToIs and 10 PoIs each, DP needs to plan 1220 collision-free paths in the second stage and spends on average 71.87 seconds in total, whereas IDP only needs 8.05 seconds on average, saving 88.80% of the planning time and is 8.9 times faster. This improvement is less noticeable in the indoor environment with 12 ToIs and 2 PoIs each, where DP preplans 48 collision-free paths spending in total an average of 7.59 seconds and IDP uses 7.20 seconds, reducing the planning time only by 5.1% due to the small number of PoI. The IRBA and IDP have roughly the same performance in terms of planning time. The differences are within 8% and 3% of standard deviation in the two environments.

		Lunar			Indoor		
		DP	IDP	IRBA	DP	IDP	IRBA
Time	Mean	71.87	**8.05**	8.25	7.59	7.20	**7.18**
	Std.	8.89	2.50	2.78	1.35	1.00	0.59
Cost	Mean	**97.22**	97.58	100.08	**82.05**	82.85	83.66
	Std.	1.16	1.83	2.33	3.75	3.70	4.02

TABLE I: Comparison of DP (brute force optimal path), IDP (ours) and IRBA (adapted from [29]). IDP generates slightly higher cost than DP because DP has a denser graph while planning all paths.

		$t_{high}=0.3$ $t_{low}=0.3$	$t_{high}=0.8$ $t_{low}=0.3$	Full Collision Checking
Time	Mean	**8.05**	9.39	11.56
	Std.	2.50	3.61	2.29
Cost	Mean	97.58	97.87	**97.17**
	Std.	1.83	1.90	1.79

TABLE II: Comparision between traversability threshold and full collision checking in the lunar environment.

IDP produces similar path costs as DP in both tests (Table I). The difference is due to the denser graph that DP constructs while planning more paths. However, IRBA increases the cost compared to DP by $2-3\%$, and does not provide any theoretical optimality guarantee compared to IDP, therefore in different environments, the performance gap IRBA and IDP may be larger. When running the experiment on the same graph, IDP produces the same path cost as DP as explained in Section IV-D, while IRBA often produces worse results.

2) Hierarchical Obstacle Avoidance: We evaluate the effectiveness of the hierarchical obstacle avoidance scheme in the lunar simulation environment (Fig. 5a). The environment contains two obstacles, 13 stones (ToIs), two craters and various slopes, resulting in various traversability and making it more challenging and realistic than the indoor scene. As shown in Table II, with the use of the traversability module, 30% of the planning time is saved due to fewer voxel queries of the state validity checker. We set the collision checking for the robot base to stop at the second iteration, as there the outer spheres are already a sufficiently good approximation of the ANYmal robot's base (Fig. 4c). The acquisition of the traversability of a state only requires one voxel query, whereas iteratively checking collision for that state requires at most four queries (when stopping at the second iteration). Traversability filtering marginally increases the geometric path cost but leads to a safer path. This is illustrated in Fig. 8;3, where the planned path with full collision-checking passes over the stone, given that the poses do not incur any collision. On the other hand, the traversability module can account for the risk and filters out the high-risk areas, avoiding passing over the stone.

The breakdown of the timings in Table III shows that the

	$t_{high}=t_{low}=0.3$	Full Collision Checking
A*	5.05	4.79
Sampling	0.43	0.84
Edge checking	**2.32**	**5.68**
Vertex checking	0.01	0.05
Miscellaneous	0.24	0.23
Total	8.05	11.56

TABLE III: Breakdown of the required time of each computation step.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

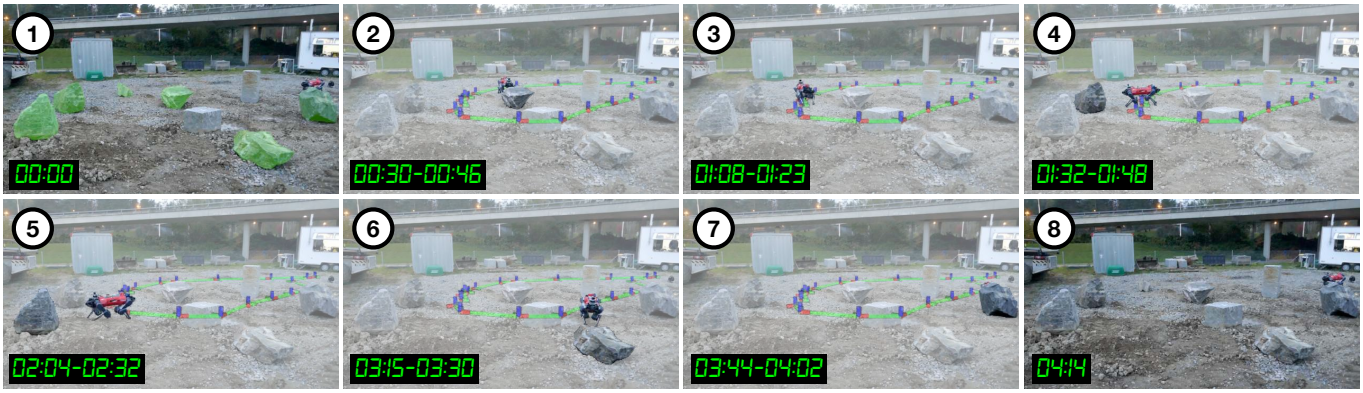


Fig. 7: Real-world deployment on the ANYmal quadruped robot. The subfigures show the snapshots of the entire mission. The six specified targets are marked in green in (1). The robot follows the global path to perform inspection near each target one by one (2-7, with the current robot inspection pose and target highlighted in a dimmed background). The time span where the robot stays at each target is given in each picture relative to the mission start, each of which includes 15 seconds for inspection and the computation time of the local planner generating the next path. Finally, the robot homes in (8).

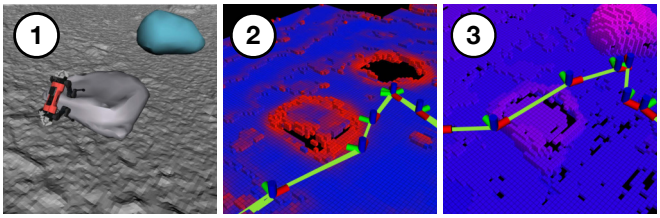


Fig. 8: (1) Risky region for the robot to traverse. (2) The path planned with the traversability filtering avoids the risky zone. (3) The path planned with only collision-checking choose to pass over it. Although there is no collision with the environment, the path is risky.

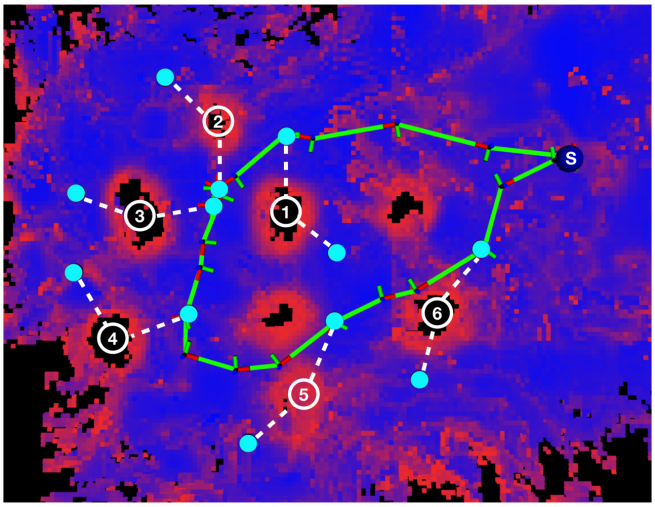


Fig. 9: Generated path (green) for the real-world experiment. The ToIs are labeled with white numbers and connected to their respective PoIs (marked as cyan dots) with white dashed lines. The robot start pose is marked as a blue dot and labeled with S.

improvement of the planning time mostly originates from edge checking, which requires a large amount of vertex checking and hence many more voxel queries if using volumetric collision checking.

3) Scalability: To show the scalability of the method, we deploy the proposed planner in the lunar environment to missions of different numbers of ToIs while keeping the PoI number of each ToI fixed at six. As shown in Table IV, the planner is able to plan through 48 ToIs within three minutes, saving 60% to 80% of planning time compared to DP while having similar path costs. The planning time of IDP increases

ToI	Time			Cost		
	DP	IDP	IRBA	DP	IDP	IRBA
6	5.42 ±0.78	1.05 ±0.22	**0.74** ±0.10	**64.55** ±0.18	65.17 ±0.54	66.93 ±0.66
12	20.12 ±1.71	**5.12** ±1.77	5.82 ±1.82	**87.75** ±0.71	89.04 ±1.38	92.04 ±2.69
24	80.49 ±8.37	22.31 ±3.07	**19.67** ±4.58	**122.55** ±0.64	123.27 ±0.72	124.10 ±1.06
48	508.71 ±55.80	**176.17** ±20.44	180.42 ±31.52	164.04 ±2.96	**163.15** ±3.00	167.55 ±2.78

TABLE IV: The average computation time [s] and path cost of DP, IDP and IRBA with their respective standard deviation when the number of ToI varies.

roughly quadratically as the number of ToI increases from 6 to 24, and more than quadratically from 24 to 48. This is because the planning time does not only depend on the number of paths but also on the difficulty of individual paths and the size of the incrementally built planning graph.

B. Real World Deployment

We deploy the planner on the ANYmal quadruped robot in a real-world outdoor multi-goal mission on pitted terrain covered by gravel (Fig. 7). Six stones with irregular shapes are specified as ToI and two cuboid-shaped stones are placed on the terrain as obstacles.

We first command the robot to walk in the environment to build the needed TSDF and traversability map. Then we select for each ToI two PoIs to choose from. The planner generates the optimal global path accordingly. Thereafter, the path segment consisting of $SE(3)$ waypoints from the current position to the next target is sent to a low-level local planner [12], which generates the local path following the waypoints. We give 15 seconds to investigate each ToI. Once the inspection at one ToI is finished, the global path to the next target is sent to the local planner. The robot is able to conduct the mission fully autonomously under the guidance of the proposed global planner, as shown in Fig. 7. The robot finishes visiting a total of six targets and homing in 4 minutes and 14 seconds. The experiment shows that the entire pipeline works in a real-world environment with rough terrain. A single minor collision with a stone happened due to the inaccurate tracking of our collision-free path (Fig. 9).

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

VI. CONCLUSIONS

In this work, we present the SMUG planner, a safe multi-goal planner that can rapidly generate an optimal global path and is applicable to various exploration, and industrial inspection missions. With the proposed IDP algorithm, the planner can generate a safe global path visiting 12 targets in 8 seconds and is 8.9 times faster than the naive approach without losing optimality. The proposed hierarchical state validity checking scheme increases safety, and reduces the total planning time by 30% compared to pure volumetric collision checking. We conduct experiments in which the ANYmal robot performs a multi-goal mission fully autonomously on rough terrain, showing the real-world applicability of the proposed planner. To the best of our knowledge, this is the first time a global planner is deployed on a mobile robot to a real-world GTSP with collision-free path planning.

We assume the ToIs can be inspected from all corresponding PoIs. However, in some real scenarios, if a ToI is not inspectable from the selected PoI, a fallback option is needed.

Since the planner is designed for ground robots and uses a robot-specific traversability map, it is not straightforward to generalize the entire method to robots with more degrees of freedom, such as drones. However, IDP is generalizable to all robots given a corresponding point-to-point planner.

Moreover, the default OMPL cost for $SE(2)$ space does not capture the traversing capabilities of the legged robot well. To counteract this we rely on a simple post-processing heuristic which aligns the waypoints heading to ensure the robot moves forward rather than sideways. A more sensible path cost, eventually learned in simulation, tailored to the motion of the legged robot and a corresponding informed sampler could solve this.

REFERENCES

[1] A. Wedler *et al.*, "Preliminary results for the multi-robot, multi-partner, multi-mission, planetary exploration analogue campaign on mount etna," in *72nd International Astronautical Congress (IAC)*, October 2021.

[2] D. Malaut, "Esa-esric space resources challenge final," October 2022.

[3] C. Gehring *et al.*, "Anymal in the field: Solving industrial inspection of an offshore hvdc platform with a quadrupedal robot," in *International Symposium on Field and Service Robotics*, 2019.

[4] *Autonomous Robot for Gas and Oil Sites*, vol. All Days of *SPE Offshore Europe Conference and Exhibition*, 09 2015.

[5] S. L. Smith and F. Imeson, "GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 87, pp. 1–19, 2017.

[6] D. L. Applegate *et al.*, *The Traveling Salesman Problem: A Computational Study*. Princeton: Princeton University Press, 2007.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[8] J. Frey *et al.*, "Locomotion policy guided traversability learning using volumetric representations of complex environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5722–5729, 2022.

[9] T. Cieslewski, , *et al.*, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2135–2142, 2017.

[10] A. Bircher *et al.*, "Receding horizon "next-best-view" planner for 3d exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1462–1468, 2016.

[11] T. Dang *et al.*, "Graph-based path planning for autonomous robotic exploration in subterranean environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3105–3112, 2019.

[12] L. Wellhausen and M. Hutter, "Rough terrain navigation for legged robots using reachability planning and template learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6914–6921, IEEE Press, 2021.

[13] L. Kavraki *et al.*, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[14] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2951–2957, 2015.

[15] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966–984, 2018.

[16] G. Laporte *et al.*, "Generalized travelling salesman problem through n sets of nodes: An integer programming approach," *INFOR: Information Systems and Operational Research*, vol. 21, no. 1, pp. 61–75, 1983.

[17] E. M. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," *Discrete Applied Mathematics*, vol. 55, no. 3, pp. 197–218, 1994.

[18] C.-M. Pintea, P. Pop, and C. Chira, "The generalized traveling salesman problem solved with ant algorithms," *J Univers Comput Sci*, vol. 13(7)rem, 08 2017.

[19] C. Noon, "A lagrangian based approach for the asymmetric generalized traveling salesman problem," *Operations Research*, vol. 39, 08 1991.

[20] C. Noon and J. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR. Information Systems and Operational Research*, vol. 31, 02 1993.

[21] J. Yang, X. Shi, M. Marchese, and Y. Liang, "Ant colony optimization method for generalized tsp problem," *Progress in Natural Science - PROG NAT SCI*, vol. 18, 11 2008.

[22] L. Snyder and M. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 174, pp. 38–53, 10 2006.

[23] I. Gentilini, *Multi-Goal Path Optimization for Robotic Systems with Redundancy based on the Traveling Salesman Problem with Neighborhoods*. PhD thesis, Carnegie Mellon University, 2012.

[24] D. Karapetyan and G. Gutin, "Lin–kernighan heuristic adaptations for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 208, no. 3, pp. 221–232, 2011.

[25] I. Gentilini, F. Margot, and K. Shimada, "The travelling salesman problem with neighbourhoods: Minlp solution," *Optimization Methods and Software*, vol. 28, no. 2, pp. 364–378, 2013.

[26] S. Alatartsev *et al.*, "On optimizing a sequence of robotic tasks," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 217–223, IEEE, 2013.

[27] C. Wurll, D. Henrich, and H. Wörn, "Multi-goal path planning for industrial robots," 1999.

[28] J. Faigl, V. Vonásek, and L. Preucil, "A multi-goal path planning for goal regions in the polygonal domain.," in *ECMR*, pp. 171–176, 2011.

[29] W. Gao *et al.*, "Automatic task scheduling optimization and collision-free path planning for multi-areas problem," *Intell. Serv. Robot.*, vol. 14, p. 583–596, sep 2021.

[30] X. Pan *et al.*, "Approximate shortest path algorithms for sequences of pairwise disjoint simple polygons," in *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry, CCCG 2010*, pp. 175–178, 2010.

[31] K. Vicencio, B. Davis, and I. Gentilini, "Multi-goal path planning based on the generalized traveling salesman problem with neighborhoods," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2985–2990, 2014.

[32] M. Kulkarni *et al.*, "Autonomous teamed exploration of subterranean environments using legged and aerial robots," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3306–3313, 2022.

[33] C. Cao, H. Zhu, H. Choset, and J. Zhang, "TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments," in *Proceedings of Robotics: Science and Systems*, (Virtual), July 2021.

[34] H. Oleynikova *et al.*, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1366–1373, 2017.

[35] I. A. Sucan *et al.*, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[36] L. Perron and V. Furnon, *OR-Tools*. Google.