



Computationally-Fast Coverage Path Planning for Planetary Multi-Modal Mobility Robots Traversing Disconnected Regions

Sebastian Stelter¹ · Sabyasachi Mondal¹ · Leonard Felicetti¹ · Saurabh Upadhyay¹

Received: 8 December 2025 / Accepted: 13 April 2026
© The Author(s) 2026

Abstract

Planetary exploration missions require efficient coverage path planning methods to reliably and rapidly investigate vast areas of scientific interest. This task is particularly challenging due to strict computational limitations and complex terrain properties, with nontraversable regions separating the target area into multiple arbitrarily shaped safe regions. For covering such terrains, the proposed work presents a multi-stage heuristic coverage path planning method that utilizes multi-modal robots capable of switching between ground and aerial locomotion. Safe regions are extracted from a Digital Elevation Map (DEM) and individually covered by solving a Traveling Salesman Problem (TSP) using a Lin-Kernighan Heuristic solver. The sub-tour endpoints are then connected through an additional TSP to generate the complete coverage path. Unlike existing approaches that assume convexity, the proposed method generalizes to complex planetary scenarios and prioritizes computational efficiency. The method is evaluated on over 500 randomly generated maps, as well as scenarios derived from Martian DEM data, using a single core of a low computation power embedded computer. The experimental results demonstrate the viability and computational efficiency of the approach across various complex scenarios with arbitrarily shaped regions. When compared with state-of-the-art planners, the proposed method achieves substantially reduced computation times while producing coverage paths of comparable length.

Keywords Coverage path planning · Planetary exploration · Planetary autonomy · Complex terrains · Traveling salesman problem · Multi-modal robots

1 Introduction

Coverage Path Planning (CPP) plays a pivotal role in modern robotics, enabling robots to navigate and survey complex environments. In the context of planetary exploration, CPP allows planetary robots to efficiently map environments, collect samples and identify points of interest. While CPP has been widely applied in domains such as agriculture or

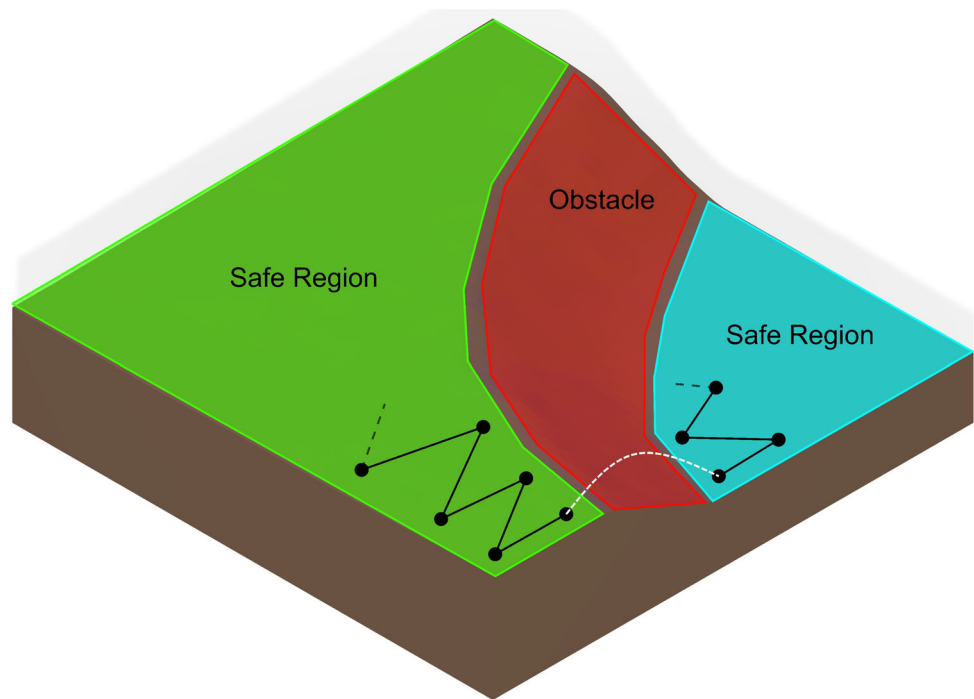
industrial inspection [1], extending these methods to planetary exploration remains difficult due to the need to plan efficiently in unstructured terrains. These environments often contain regions with obstacles such as cliffs, boulders or steep inclines [2], which ground rovers cannot traverse safely due to their mechanical capabilities. We define a safe region as any portion of the terrain that is free of such obstacles and therefore considered suitable for robotic traversal. Due to the placement of obstacles, safe regions can take arbitrary shapes, including convex and non-convex forms. Safe regions may also fully enclose one or multiple obstacles. In many scenarios, obstacles can entirely separate safe regions from each other, as shown in Fig. 1.

While ground rovers are incapable of completely covering these maps, aerial robots could avoid obstacles by flying over them. However, aerial robots suffer from limited onboard resources, limited range, higher risk of failure, higher energy consumption, and reliance on a suitable atmosphere. Further, due to the lower vantage point, ground-based rovers can get a better view of narrow or covered regions.

✉ Sebastian Stelter
sebastian.stelter@cranfield.ac.uk
Sabyasachi Mondal
sabyasachi.mondal@cranfield.ac.uk
Leonard Felicetti
leonard.felicetti@cranfield.ac.uk
Saurabh Upadhyay
saurabh.upadhyay@cranfield.ac.uk

¹ Faculty of Engineering and Applied Sciences, Cranfield University, Cranfield MK430AL, Bedfordshire, United Kingdom

Fig. 1 Illustration of the Multi-Modal Multiple Disconnected Regions CPP (M3DR-CPP) problem near a crater edge on Mars. The robot must travel from the green region over the red obstacle into the blue region using its aerial modality to cover the entire map. Black lines show the coverage path between the black waypoints; the white dashed line shows the use of the aerial modality. DEM data courtesy of MarsTrek [63]



To bridge the gap between ground rovers and aerial robots, and to take advantage of the benefits of both platforms, multi-modal mobility robot platforms have been proposed for planetary missions [3, 4]. These robot platforms are capable of changing their locomotion modalities between ground locomotion (e.g., driving or walking) and aerial locomotion (e.g., jumping or flying). These platforms could utilize their ground based modality to safely and efficiently carry out coverage tasks in safe regions, whereas the aerial modality could be used to traverse between regions, thus ensuring complete coverage while minimizing the use of the aerial modality to reduce risk and energy consumption. However, despite advances in the development and control of multi-modal robots, planning methods remain comparatively underexplored.

One of the key challenges lies in generating solution paths for multi-modal robots and determining modality transition points while remaining computationally feasible under the strict hardware constraints of planetary robots. Due to the reliance on lightweight and radiation tolerant processors, these systems face severe processing limitations [5]. To the best of our knowledge, the approach proposed in this paper is the first to address this challenge by enabling coverage path planning in a multi-region planetary exploration scenario with arbitrarily shaped regions while maintaining a low computational overhead. In contrast to prior work, which often depends on rich sensor suites, the proposed method relies solely on Digital Elevation Map (DEM) data.

This work considers a specific class of CPP problems where a multi-modal locomotion robot is used to cover a

target region on a planetary surface, consisting of multiple disconnected safe regions and obstacles. Unlike existing earth based work, which primarily considers convex regions [6], the proposed approach generalizes to arbitrarily shaped regions, including both convex and non-convex shapes, which may also fully contain obstacles.

This scenario is formulated as a specific use-case of the Multi-Modal Multiple Disconnected Regions CPP (M3DR-CPP) problem as defined in our earlier work [7]: Given the DEM of the area and a multi-modal mobility robot as described above, the goal is to generate a minimum length coverage path that covers and connects all safe regions. Obstacles can completely separate the area into multiple disconnected regions and can only be crossed by using an alternative locomotion modality without performing coverage operations.

In this work, we revisit the M3DR-CPP problem, focusing primarily on the aspect of computational efficiency. We propose a computationally fast hierarchical approach to solve the M3DR-CPP problem by first generating a near-optimal coverage path within each safe region before connecting them. The approach formulates a Traveling Salesman Problem (TSP) for each disconnected region and solves it using a Lin-Kernighan Heuristic (LKH) based solver [8, 9]. The individual intra-region coverage paths are then connected by formulating and solving an additional inter-region TSP using the same approach.

The viability of the proposed approach is highlighted with experimental validation on randomly generated scenarios, as well as scenarios using realistic DEM data of the Martian

surface. These experiments are performed on a Raspberry Pi 4B to demonstrate the low computational demands of the method. This platform was selected due to its prior use in space missions [10], its proposed application in planetary surface operations [11], and its evaluation by NASA as a cost-effective alternative to conventional space-qualified systems [12]. Comparative analysis with state-of-the-art multi-region CPP solvers further highlight the computational efficiency of the proposed method and its robustness in handling scenarios with irregularly shaped regions.

In particular, the following contributions are made in this work:

- A novel coverage path planning method is proposed for multimodal robots in complex planetary terrains which generates efficient coverage paths in scenarios with multiple disconnected arbitrarily shaped regions.
- A hierarchical strategy is developed which significantly reduces calculation times while preserving solution quality.
- Experimental studies are carried out to demonstrate applicability in various complex scenarios and on platforms with limited computational resources.

The remainder of this paper is structured as follows. Section 2 discusses previous work in the area of CPP. Section 3 gives a detailed description of the M3DR-CPP problem. Section 4 describes our proposed approach and Section 5 presents the experimental and comparison analysis. Finally, Section 6 concludes the work.

2 Related Work

This Section reviews the existing works on the use of multimodal robots in space missions, followed by methods for identifying safe and unsafe terrain. At the end, coverage path planning techniques for both single-region and multi-region settings are reviewed.

2.1 Multi-Modal Robots

Multi-modal robot platforms have been suggested to traverse complex terrains in a variety of scenarios. These robots typically alternate between either driving and flying, or driving and jumping. Ground based locomotion is generally more energy efficient and therefore favored in scenarios with limited battery capacity, while aerial locomotion is preferred when considering spatial constraints [13]. Planetary surface applications generally consider a jumping robot as it does not rely on an atmosphere. The jumping robot models proposed for planetary missions usually take the shape of spherical robots [4] or tensegrity robots [3, 14]. These designs offer

a strong protection against the extreme conditions on planetary surfaces, as well as a high resilience against collisions, making them strong candidates for these exploration missions. Although the range of jumping robots is significantly shorter than that of flying robots, low gravity environments such as the Moon allow jumping robots to cover large distances efficiently [14]. Multiple consecutive jumps can also be used to reach a distant target position [15]. In other areas, wheeled [16] and legged [17] jumping robots have also been proposed.

Robot platforms that can drive and fly usually have wheels attached to the outside of a quadcopter setup [18] or individual rotors attached to each wheel [19]. While their applicability is limited to planets with dense atmospheres, their long range makes them strong candidates for large scale exploration scenarios. The Ingenuity robot has also shown that flight in comparatively thin atmospheres is feasible [20], extending the environments in which these robots can operate.

2.2 Safe Region Detection

When navigating unstructured terrains, identifying traversable regions is essential to prevent robots from becoming immobilized or damaged. DEMs are frequently employed to distinguish between safe and unsafe regions. In the Hayabusa2 mission, potential landing sites were detected by calculating the slopes of different surface elements, among other factors [21]. Safe regions on asteroid surfaces have been identified by fitting planes through elevation grids and computing their slopes [22]. DEM data has also been used to predict slippage and maximum traversal speeds, enabling pre-computation of multiple viable paths to reach a target in Mars sample return scenarios [23]. In addition, DEMs have been used to augment local perception, for example by providing slope information to enhance camera-based semantic segmentation [24] or by supplying absolute height data to complement locally computed motion costs [25].

In environments where DEM data is not available, traversable regions are typically inferred from local terrain representations constructed using onboard sensors. Elevation maps that encode factors such as slope, roughness and obstacle proximity have been used to assign a risk factor to each grid cell for risk-aware path planning [26]. Traversal risk has also been quantified by computing and analyzing the normal vectors of terrain mesh faces [27].

2.3 Coverage Path Planning

The CPP problem has been explored across various scenarios, including planetary exploration, agriculture, industry, and environmental monitoring [1]. In the context of planetary exploration, most existing methods adopt an online coverage

strategy. K-means clustering has been used to partition the area into multiple segments for each robot [28]. Each robot then explores its own segment using a frontier-based method. The partitioning is periodically recalculated to balance workload and account for robot failures. Instead of creating fixed region assignments, a shared sparse graph has been used to collaboratively cover an area [29]. The robots choose frontier nodes on the graph based on proximity and volumetric gain. In a single robot exploration scenario, DEM data was combined with local observations to guide the coverage [25]. A gaussian process has been used to specifically target information rich areas during exploration to create a model of the water distribution in a lunar crater [30]. These approaches enable adaptive responses to environmental changes and evolving mission objectives. However, lack of prior knowledge prevents them from ensuring complete coverage and increases the risk of encountering local extrema, leading to coverage path overlap.

In general, existing CPP works focus on 2D scenarios containing a single connected safe region and a limited number of obstacles. They can be classified into classical and heuristic approaches [1]. Classic approaches, such as Boustrophedon coverage [31, 32] or spanning tree coverage [33, 34] use a structured pattern to systematically explore the area. Although these approaches are computationally simple, they generate suboptimal paths in non-convex regions. Boustrophedon algorithms use cellular decomposition to separate a non-convex area into multiple convex cells that are covered individually. As the path for each cell is generated independently, transitions between cells are often significantly longer than required any may exceed the region boundaries. This problem was partially addressed by allowing for several entry and exit points per region and generating Boustrophedon paths based on the chosen entries [35]. Adding multiple potential entry points reduces the overall path length, but significantly increases the search space, requiring the use of a neural network to generate the coverage path. Boustrophedon paths and spiral coverage paths have been proposed for planetary exploration missions [36]. Whereas Boustrophedon paths are suitable for exploration using short range sensors, spiral paths can be used in combination with long range sensors by tracking the position of a retroreflector at the center of the spiral. Similarly, square spiral paths have been proposed for a multi-robot system exploring a planetary surface [37]. While these approaches allow for easy localization of the robot, spiral paths cannot easily adapt to non-convex or cluttered regions. Spanning tree coverage adapts to non-convex regions more efficiently, but the reliance on large grid cells to surround the spanning tree can cause large areas of free space to remain uncovered. Heuristic approaches use stochastic search strategies to reduce problem complexity and hence computation time. However, these methods do not guarantee an optimal coverage path. Common heuris-

tic approaches include the use of genetic algorithms [38] or neural networks [39]. Genetic algorithms have been used in combination with Boustrophedon paths to determine the optimal orientation of the sweep lines in 3D terrains [40]. A different heuristic was used to grow individual path segments based on slopes in a 3D environment [41]. The task of connecting the path segments was then phrased as a TSP and solved using a genetic algorithm. The LKH [8, 42] has also been used to generate coverage paths by formulating the CPP problem as a TSP [43, 44]. In benchmarks using the TSPLIB dataset [45], the LKH found the exact optimal solution for 101 out of 104 problems with known optima, and improved upon the best known solution in 6 out of 7 problems without known optima [9]. For large-scale problems, it surpasses other state-of-the-art solvers in accuracy while maintaining comparable computation times [46].

CPP approaches in disconnected regions typically assume that the region boundaries can be safely traversed without risk to the robot, for example in Unmanned Aerial Vehicle (UAV) monitoring scenarios [6, 47]. The Boustrophedon algorithm has been used to generate a simple line-sweeping path in each region, which are then connected using a variety of different methods, such as simulated annealing [48], generalized large neighbourhood search [47] or genetic algorithms [49]. By calculating multiple possible Boustrophedon paths per region and using a heuristic solver, a near-optimal coverage path was achieved [6]. The solver alternates between modifying the region order and updating the Boustrophedon paths to gradually improve the path length. Similarly, coverage paths have been generated by calculating multiple Boustrophedon paths per region, but using a k-nearest neighbour approach to determine the inter-region path [50]. Spanning tree coverage has been used in combination with ant colony optimization to cover multiple regions using a robot team [51]. Each robot was assigned regions based on priority constraints and determined the region visit order using ant colony optimization. Spanning tree coverage was then used to generate a path within each region. While the above approaches are capable of generating coverage paths in multi-region scenarios, they all assume all regions to be convex and obstacle free.

Convex regions with obstacles have also been considered. A bat algorithm has been used to avoid obstacles that are encountered while following a Boustrophedon path [52]. Inter-region paths were then generated using a neural network. Similarly, obstacles have been avoided by flying evasive maneuvers around or over them, however, using a set of sensor footprint sweeps to generate the intra region path instead [53]. Ant colony optimization was then used to connect the regions and generate the complete coverage path. While these approaches are capable of avoiding obstacles within the region, the general shape of each region is still expected to be convex and the obstacle avoidance maneuvers cause a significant overlap near the obstacles.

Non-convex regions have been included using a variety of strategies. By calculating the convex hull of non-convex regions, Boustrophedon paths could be calculated similarly to the above strategies [54]. A genetic algorithm was then used to generate inter-region paths. However, by using the convex hull, the coverage path length significantly increased. Moreover, it is unsuitable for planetary surface exploration, as extending beyond region boundaries would require traversing unsafe terrain. Alternatively, convex decomposition has been used to cover multiple non-convex regions [55]. 3D target areas were flattened into 2D space and decomposed into convex cells. Boustrophedon paths were then generated for each cell and the resulting path was projected back into 3D space to create the coverage path. However, the connection between the convex cells may cross region boundaries and increase the path length. A headland path has been used to guide the generation of Boustrophedon paths in non-convex regions [56]. This allowed the robot to fully stay within the region boundary, but causes a significant path overlap, as the entire region outline was covered twice. Further, since a nearest-neighbour heuristic was used to determine the region visit order, the inter-region path was also significantly longer than required. A system of concentric rings has been used to generate a coverage path within convex and non-convex regions [57]. Each region was first filled with concentric boundary following rings. Self-ordering maps were used to identify the region visiting order and dynamic programming was used to detect the optimal transition point between rings. While this approach is capable of covering multiple non-convex regions without exceeding region boundaries, it has a high computational complexity and may perform poorly in obstacle rich environments.

Although substantial advances have been made in the field of CPP, particularly for scenarios with disconnected regions, some key challenges persist. Especially the generation of coverage paths in complex terrains with arbitrarily shaped regions and high obstacle density remains challenging. Additionally, the strict computational requirements of planetary robotics are largely unaddressed, restricting the applicability of most existing methods in these contexts.

3 Problem Description

We consider a multi-modal robot with ground and aerial modalities, covering a target area on an unstructured planetary surface. The robot receives a DEM of the target area from a satellite, such as the Lunar Reconnaissance Orbiter carrying the Lunar Orbiter Laser Altimeter [58]. The DEM is processed as a triangular mesh with a triangle size set such that the sensor footprint of the robot entirely covers the triangle area from its centroid. Triangles steeper than the user-defined slope threshold are treated as obstacles, while

those at or below the limit are treated as safe regions. Obstacles and safe regions can be of arbitrary shape. Obstacles may entirely disconnect safe regions from each other, as shown in Fig. 1, and may be entirely enclosed by safe regions.

While using the ground modality, the robot can carry out the coverage task but is restricted to moving through safe triangles. In the aerial modality, it can fly over obstacles, but it cannot perform the coverage task. The robot can rotate in place without needing a turning radius. While using the aerial modality, it can move between any two points on the map without landing. Hence, straight line paths are considered between any two waypoints.

Under these constraints, we can describe the problem as a special case of the M3DR-CPP problem [7] as follows:

1.: Given the DEM of the target, identify safe regions and obstacles. Discretize the safe regions into waypoints that ensure complete coverage.

2.: Given a set of regions with waypoints, and a set of obstacles, find a coverage path that passes through all waypoints in all safe regions exactly once while minimizing the path length and the use of the aerial modality. The generated path must avoid all obstacles by either driving around them or using the aerial modality to traverse them.

4 Proposed Coverage Path Planning Method

The proposed method addresses the M3DR-CPP problem by solving the two subproblems as shown in Fig. 2. To solve the first subproblem, a triangular mesh is modeled from the DEM data and safe and obstacle regions are detected. For each safe region, waypoints are generated based on the centroids of the mesh triangles. The second subproblem is addressed by solving multiple smaller subproblems to reduce computational complexity. First, an open loop TSP is solved for each disconnected region and the two endpoints of the coverage path are determined. Then, a TSP is solved using the region endpoints of the previous step to connect all regions and generate a near-optimal coverage path. The approaches to each subproblem are described below.

4.1 Terrain Modeling and Region Detection

The region detection progress is detailed in Algorithm 1 and shown in the upper half of Fig. 2. To model the terrain, the DEM of the target area is first imported as a triangulated mesh (line 1). By controlling the triangle size of the input mesh, sensor footprints can be modeled. Alternatively, if sensor footprints are not a critical factor, the Lindstrom-Turk edge collapse strategy [59, 60] can be applied to reduce the number of triangles below a threshold to allow faster computation.

We then adopt the obstacle and region detection strategy from our previous work [7]. The angle of each triangle normal

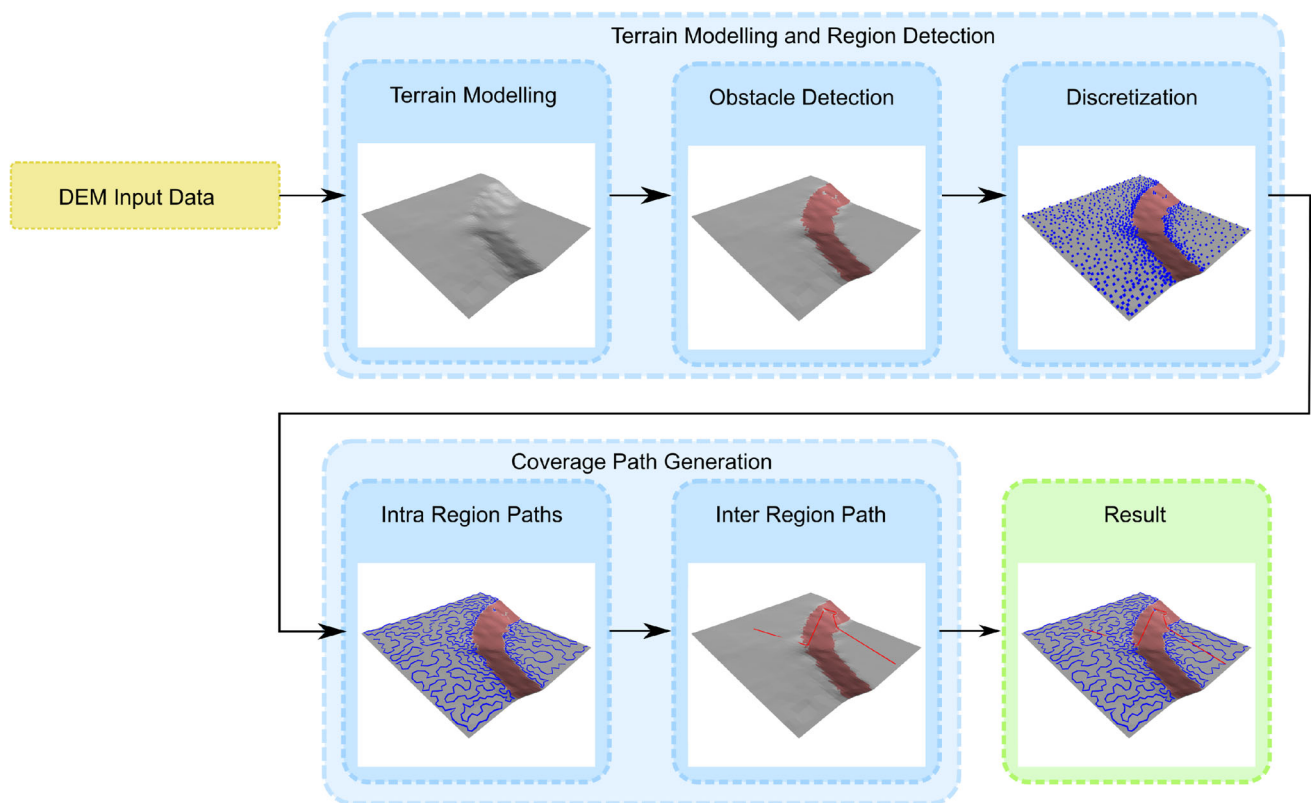


Fig. 2 Flowchart of the proposed algorithm. The data received from the DEM is analyzed to determine safe regions. Disconnected regions are detected, and the centroids of each triangle are calculated as waypoints. A TSP is then solved in each region and between all regions to generate a full coverage path

vector is compared against a slope threshold to determine safe areas and obstacles (lines 3–9). If the calculated slope exceeds the slope threshold, the triangle is considered as an obstacle (line 5), otherwise it is considered as a safe triangle (line 7). By conducting a breadth first search over all safe triangles, disconnected safe regions are detected (line 10). Triangles that only share a vertex are not considered to be in the same region, as there is no safe way to transition between them.

Finally, each safe region is discretized by calculating the centroids of all safe triangles (lines 11–14). The centroids serve as waypoints for the intra-region TSP, which determines the coverage path for that region.

4.2 Coverage Path Generation

After defining the regions and waypoints to be covered, the CPP can be represented as a series of TSPs, solved in two stages (see lower half of Fig. 2). In the first stage, each disconnected region is handled independently by formulating and solving a TSP that produces its intra-region coverage path. In the second stage, the endpoints of these intra-region paths are treated as waypoints in a separate TSP to determine the inter-region path. The resulting paths are then combined

to produce the complete coverage path. Since obstacles are considered during the creation of the intra-region paths, these segments of the coverage path can be traversed using the ground modality. The inter-region paths cross region boundaries and therefore require the use of the aerial modality.

For every region, given a set of waypoints $P = \{p_1, \dots, p_n\}$, the TSP can be defined as finding a permutation $T = (p_i, \dots, p_j)$, called a tour, that minimizes the cost function

$$c(T) = \sum_{i=1}^n d(t_i, t_{i+1}), \quad t_{n+1} \equiv t_1 \quad (1)$$

where $d(t_i, t_{i+1})$ is the distance between two subsequent waypoints in the tour. As the tour is required to be a permutation of P , it is ensured that each waypoint in P is visited exactly once.

We use the Lin-Kernighan-Helsgaund (LKH-3) [42] solver as an extension of the LKH [8, 9] to find a near-optimal path. The LKH is chosen as it is widely recognized as one of the most efficient TSP solvers, consistently finding near-optimal solutions even for complex and large-scale problems.

The proposed approach employs the LKH-3 solver in two steps to generate coverage paths as described in Algorithm 2. After the regions have been detected (line 1), each

Algorithm 1 RegionDetection.**Input:** $DEM, threshold$

Output: Waypoint sets for each region $P = \{P_1, \dots, P_n\}$, Obstacles O

```

1:  $T \leftarrow triangulation(DEM)$ 
2:  $S, O \leftarrow \emptyset$ 
3: for  $t \in T$  do
4:   if  $slope(t) > threshold$  then
5:      $O \leftarrow O \cup t$ 
6:   else
7:      $S \leftarrow S \cup t$ 
8:   end if
9: end for
10:  $R = BFS(S)$   $\triangleright R = \{R_1, R_2, \dots\}$ 
11: for  $R_i \in R$  do
12:    $P_i \leftarrow centroids(R_i)$ 
13:    $P \leftarrow P \cup P_i$   $\triangleright P = \{P_1, P_2, \dots\}$ 
14: end for
15: return  $P, O$ 
```

Algorithm 2 GenerateCoveragePath.**Input:** DEM **Output:** Tour $T_{complete}$

```

1:  $P, O \leftarrow RegionDetection(DEM)$ 
2:  $E \leftarrow \emptyset$ 
3: for  $P_i \in P$  do
4:    $P_i \leftarrow P_i \cup p_0$ 
5:    $D_i \leftarrow CalculateDistanceMatrix(P_i, O)$ 
6:    $T_i \leftarrow LKH(P_i, D_i)$ 
7:   while  $first(T_i) \neq p_0$  do
8:      $T_i \leftarrow shift\_left(T_i)$ 
9:   end while
10:   $T_i \leftarrow T_i \setminus \{p_0\}$ 
11:   $E \leftarrow E \cup \{first(T_i), last(T_i)\}$ 
12:   $T \leftarrow T \cup T_i$ 
13: end for
14:  $E \leftarrow E \cup p_0$ 
15:  $D_e \leftarrow DistanceMatrix(E, \emptyset)$ 
16:  $Constraints \leftarrow \{(e_1, e_2), (e_3, e_4), \dots\}$ 
17:  $T_e \leftarrow LKH(E, D_e, Constraints)$ 
18: while  $first(T_e) \neq p_0$  do
19:    $T_e \leftarrow shift\_left(T_e)$ 
20: end while
21:  $T_e \leftarrow T_e \setminus \{p_0\}$ 
22: for  $c \in Constraints$  do
23:    $T_e \leftarrow T_e \setminus c \cup T_c$ 
24: end for
25:  $T_{complete} \leftarrow T_e$ 
26: return  $T_{complete}$ 
```

disconnected region is treated as a separate TSP instance to be solved (lines 3–13). As each region contains only a part of the overall coverage path, it is not desirable for the robot to return to the starting point of each region before proceeding to the next one. Requiring such a return would increase the total path length and introduce additional complexity. Moreover, otherwise viable intra-region solutions would be discarded if they cannot generate a valid return path. To prevent this issue, a virtual waypoint p_0 is first added to the set of waypoints of each region (line 4) [61]. The waypoint p_0 is defined with a

distance of $d(p_0, p_i) = 0$ for all waypoints p_i . This implies that the cost to reach p_0 from any other waypoint is always 0. Thus, any edge in the solution can be substituted with a connection to p_0 without increasing path length. As the LKH seeks to minimize total tour length, it typically substitutes one of the longest edges in the tour with two zero-cost edges connected to p_0 . This additional waypoint and all edges connected to it will be deleted after the TSP is solved, leaving the two waypoints that were connected to it with only a single neighbour and creating an open TSP.

Next, the distance matrix of the region is calculated (line 5). The matrix is generated by first calculating the Euclidean distance between every pair of waypoints in the region. For every pair, it is then checked whether the edge between the two waypoints would intersect with any obstacle triangle using a divide-and-conquer approach [62]. If an intersection is found, the edge weight is set to a fixed penalty. The penalty must be sufficiently large to ensure that the cost of the penalized edge exceeds the longest possible unpenalized path in the region. This ensures a valid move exists, which reduces the path length by removing the penalized edge. Since the LKH algorithm only terminates if no such move exists, the penalized edge cannot be part of the solution. Using this distance matrix, the LKH algorithm is now invoked (line 6), returning the tour of the region. As the returned tour is a closed loop with p_0 at an arbitrary position, p_0 needs to be shifted to the front (lines 7–9) before it can be removed (line 10) to maintain the correct order. The two endpoints of the region path are recorded (line 11), as they will be used to connect all region paths.

After finding the paths for the individual regions, the endpoints of all regions need to be connected in sequence to generate a complete path over all regions. To do so, the same TSP approach as described above is used. We again add a virtual waypoint p_0 (line 14) that is later removed (lines 18–21) to generate an open loop solution as our use case does not require a direct return to the origin, but a closed loop can easily be generated by omitting this waypoint. As the aerial modality will be used to travel between regions, the distance matrix is generated without considering the obstacle triangles (line 15). To generate a valid path, it needs to be ensured that the path segment that enters a region R_k through an endpoint p_i continues to the second endpoint p_j of the same region R_k . Therefore, a constraint is added for each region, requiring the edge between the two endpoints of the same region to be included in the solution (line 16). These edges are then removed from the solution and replaced with the intra-region paths from the previous step (line 22–24).

To improve the performance of the method, the number of runs the LKH-3 solver uses is modified. The LKH-3 solver uses multiple runs with different starting configurations to avoid local optima. However, the execution time increases linearly with the number of runs. It is therefore beneficial to

select the smallest number of runs that still find the optimal path. It was experimentally determined that minimal improvement was found after 3 runs in a test set of 25 randomly generated maps of different sizes and therefore the number of runs was set to 3.

5 Simulation Results

The proposed approach was validated in different simulation scenarios to analyze computational efficiency and compare it with state-of-the-art CPP approaches. First, the proposed approach is evaluated on 500 randomly generated maps to assess its adaptability and performance. Subsequently, the approach is compared with a global TSP approach, a MILP based solver [7] and the TSP-CPP solver [6] to demonstrate computational efficiency and the ability to handle arbitrarily shaped regions. Finally, the proposed approach is compared against the other approaches on realistic scenarios generated from Martian DEM data to verify real world applicability, as well as on large randomly generated maps to analyze scalability.

All experiments are conducted on a Raspberry Pi 4B with an ARM v8 @ 1.8 GHz processor and 8 GB of LPDDR4-3200 SDRAM, except for the comparison with TSP-CPP and the MILP-based solver. The code is executed on a single core to simulate the low-spec requirements of planetary robots [5]. The proposed method is implemented in C++ using the ROS2 Jazzy framework to ensure compatibility with most modern robot platforms. A summary of all results can be found in Appendix A.

5.1 Scenario with Randomly Generated Maps

The approach is validated on 500 randomly generated maps consisting of 25×25 evenly spaced squares of two triangles each, resulting in an area of 625 m^2 . The slope of each triangle is generated randomly based on a maximum slope constraint. While the slope detection threshold $\bar{\theta}$ of our approach is set to 15° in these experiments, in each trial the maximum generated slope is set to a random value between 20° and 45° . Maximum slopes below 20° generate close to no obstacles while maximum slopes above 45° generate almost exclusively obstacles. As both of these scenarios are trivial to solve, they are excluded. To simulate more realistic slope profiles, Laplacian smoothing is used with a random smoothing factor between 0 and 1. These settings result in an average of 663.39 safe waypoints per map with a maximum of 1233, minimum of 78 and standard deviation of 302.42. The average number of disconnected regions is 49.46 with a maximum of 122, a minimum of 1 and a standard deviation of 34.94.

The proposed method is able to find valid solutions to all 500 trials. On average, the path length is 527.85 m with

a minimum of 140.55 m, a maximum of 775.01 m and a standard deviation of 161.68 m. On average, 26.45% of the total length is spent on inter-region travel and 73.55% on intra-region travel.

Figure 3 shows a selection of solutions from this simulation using our proposed method. It can be seen that this approach finds valid solutions in a variety of unstructured terrains. Safe regions are marked in light gray, whereas obstacles are marked in light red. Blue lines represent intra-region paths and red lines represent inter-region paths. Figure 3a shows the sample with the highest number of waypoints, consisting of one large and one small region with few obstacles. Figure 3b shows the sample with the highest number of regions. The large number of obstacles separates the map into many small regions. Figure 3c and d both show examples with medium-sized regions. While Fig. 3c features regions consisting of narrow safe corridors, Fig. 3d features regions that fully enclose obstacles.

Finding a solution in the above scenarios takes 54.16 seconds on average using the Raspberry Pi. The fastest trial finished in 0.96 seconds and the slowest in 393.53 seconds. The standard deviation is 64.97 seconds. Except for five trials, all other trials ended in four minutes or less. The five trials with extended computing times all contain a very large number of waypoints and features such as narrow passages and complex obstacle placements. Given that such a map presents many possible paths, but only a limited number of paths that successfully avoid all obstacles, a longer execution time is expected.

5.2 Comparative Analysis

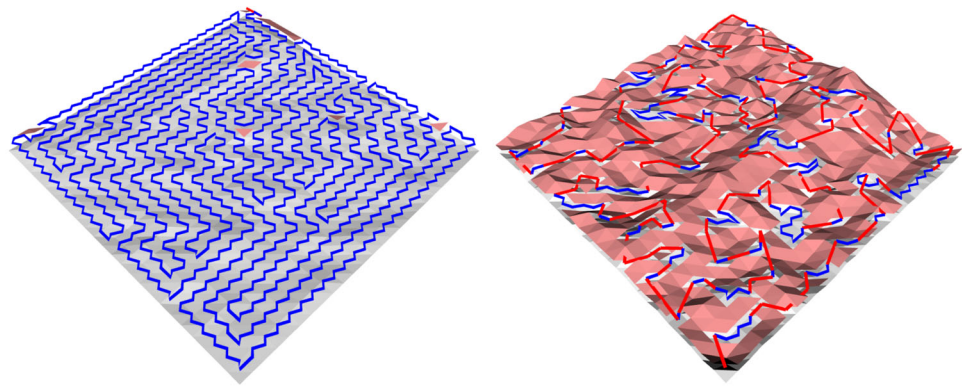
The proposed approach was evaluated against three state-of-the-art CPP solvers. First, it was compared against a global TSP method, followed by a comparison with a MILP-based solver [7] and a comparison with the TSP-CPP solver [6].

5.2.1 Comparison with Global TSP

An alternative to the proposed approach formulates the problem as a single global TSP. In this approach, all waypoints of all disconnected regions are added to a single TSP formulation with edges between any pair of waypoints. Edges that cross through obstacles have a fixed penalty added to the edge distance to minimize the use of the aerial modality.

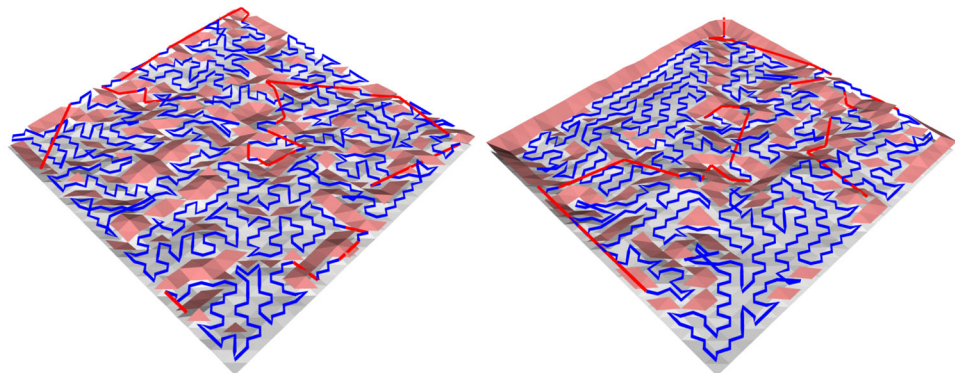
To evaluate the performance of our proposed approach compared to a global TSP formulation, the 500 randomly generated maps described in Section 5.1 have been solved using a global TSP. To ensure a fair comparison, only the problem formulation was changed as described above, using the same LKH solver and the same settings as our proposed approach.

Fig. 3 Samples from the proposed algorithm run on generated maps. Obstacles are shaded in light red on the terrain; grey regions are safe. Blue lines indicate the path taken inside of a region, while red lines indicate the use of the aerial modality



(a) waypoints: 1233, regions: 2
cost: 761.55 m, time: 112.96 s

(b) waypoints: 288, regions: 122
cost: 321.70 m, time: 4.44 s



(c) waypoints: 845, regions: 26
cost: 679.43 m, time: 17.59 s

(d) waypoints: 826, regions: 22
cost: 635.84 m, time: 46.50 s

The results of the experiment are shown in Fig. 4. On average, the proposed approach solved a problem 40.08% faster than global TSP with a standard deviation of 44.13%. While the calculation time is on average significantly shorter when using the proposed approach, it is not guaranteed that the calculation time is always shorter. In the best case, the calculation time was reduced by 91.63%, in the worst case it was increased by 170.21%. A trend can be seen that indicates a consistently smaller calculation time for small and medium sized problem (Compare Fig. 5a). However, with larger region sizes, the calculation times become increasingly unpredictable, as the problem complexity can vary strongly. Out of the 500 randomly generated maps, the proposed approach achieves a shorter calculation time than the global TSP in 412 maps.

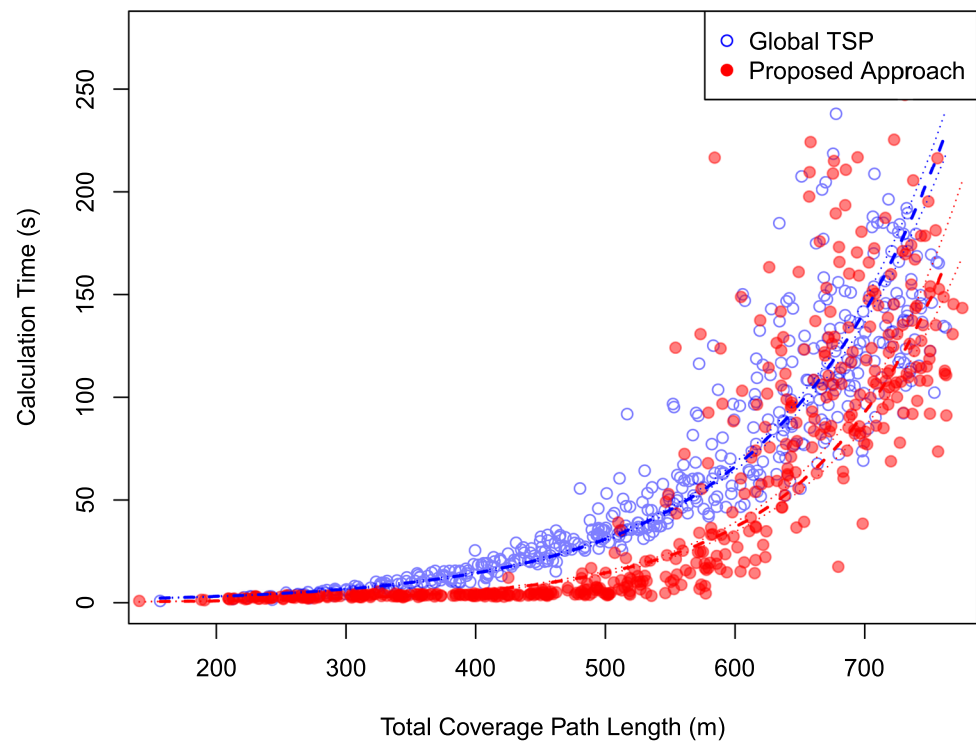
By reducing the search space, the proposed approach can also improve the path length compared to global TSP (Compare Fig. 5b). On average, the path generated by the proposed approach is 3.31% shorter than the path generated by global TSP with a standard deviation of 5.72%. In the worst case, the path generated by the proposed approach was 4.84% longer than the path generated by global TSP. In the best case, it was

28.82% shorter. It can be seen, that the proposed approach yields significant improvements in small and medium sized problems, but generates slightly worse solutions in large sized problems. Out of the 500 randomly generated maps, the proposed approach generates a shorter path than global TSP in 300 maps.

5.2.2 Comparison with MILP-based Approach

To further validate the performance of the proposed approach, it was compared with the two-stage Mixed-Integer Linear Programming (MILP) approach proposed in our previous work [7]. As the MILP approach is incompatible with the Raspberry Pi platform, this experiment was executed on a personal computer with an AMD Ryzen 7 PRO 7840U and 32 GB LPDDR5-6400 RAM. To ensure a manageable computation time, 100 smaller maps were randomly generated, following the specifications of the previous paper. Each generated map consisted of 15×15 evenly spaced squares of two triangles each, resulting in an area of 225 m^2 . The maximum slope of each map is set to a random value between 20° and

Fig. 4 Calculation times relative to path length of the calculated solution for the Global TSP approach (blue) and the proposed approach (red). A dashed regression line and dotted 95% confidence interval are drawn for each approach



40° and the smoothing factor between 0.0 and 0.7. These settings result in an average of 194.46 waypoints per map with a maximum of 369, minimum of 38 and standard deviation of 80.80. The average number of disconnected regions was 27.50 with a maximum of 49, minimum of 2 and standard deviation of 11.82.

To ensure a fair comparison between both approaches, the approach proposed in this paper has been ported to MATLAB, such that both approaches use the same programming language. While the core algorithm is identical between the MATLAB and c++ versions, calculating intersections with obstacles is significantly more efficient in the c++ version. To account for this, the calculation times of the c++ implementation are also mentioned below.

The path lengths generated by both approaches are almost identical with an average of 171.65 for the MILP approach and an average of 171.69 for the proposed approach. The optimal solution was found by both approaches in 92 out of 100 maps. In the remaining 8 maps, the path generated by the proposed approach was slightly longer than the optimal MILP path, with the worst case being 1.21% longer than the MILP path.

In terms of computation time, the MILP based approach took an average of 36.01 seconds, with a maximum runtime of 602.91 seconds, minimum of 1.88 seconds and standard deviation of 83.09 seconds. The MATLAB implementation of the proposed approach took an average of 3.93 seconds, with a maximum of 12.15 seconds, minimum of 1.56

Fig. 5 Relative difference in calculation time (a) and path length (b) depending on the total number of waypoints. Negative numbers indicate the proposed performed better than Global TSP

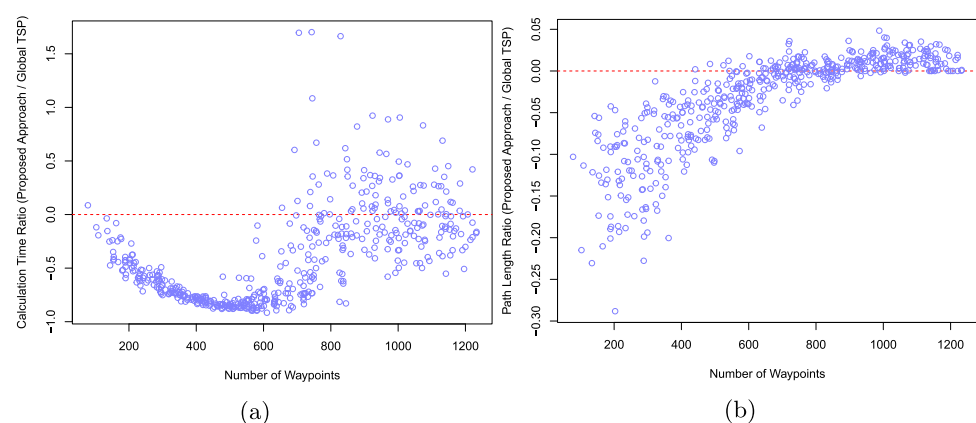
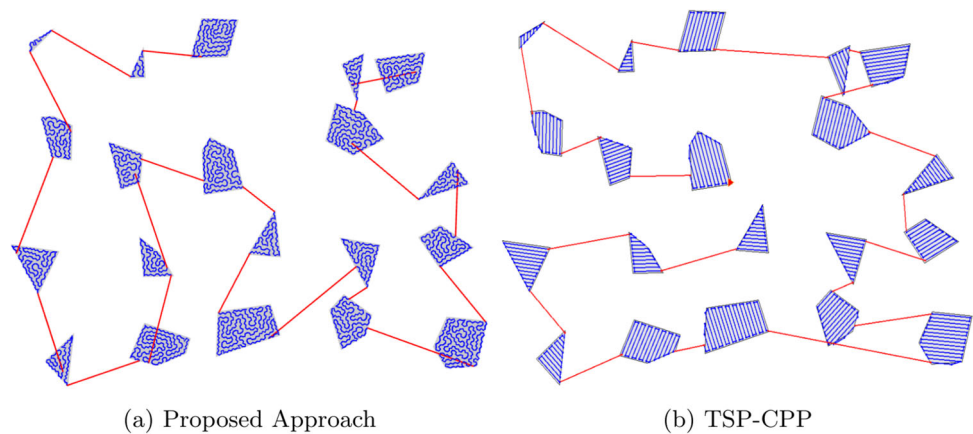


Fig. 6 Comparison between the proposed method and the TSP-CPP method in a scenario with twenty convex regions



seconds and standard deviation of 2.56 seconds. The c++ implementation of the proposed approach took an average of 0.57 seconds, with a maximum of 4.72 seconds, minimum of 0.04 seconds and standard deviation of 0.94 seconds.

5.2.3 Comparison with TSP-CPP

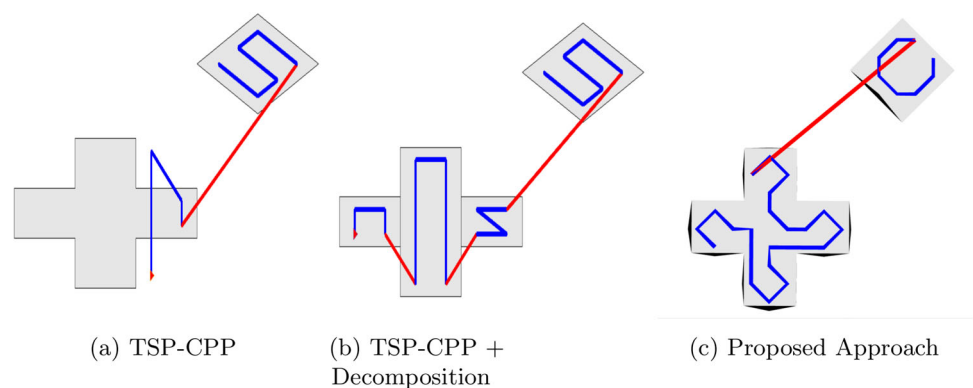
Four scenarios containing convex polygons and one scenario containing arbitrarily shaped polygons are used for comparison with the TSP-CPP solver [6]. The TSP-CPP solver is set to use a sensor footprint of 2×2 m. To ensure comparability, for our proposed approach the terrains are tessellated with a maximum triangle area of 4 m^2 .

Four of the scenarios described in the TSP-CPP paper [6] are evaluated containing 2, 20, 50 and 100 convex regions, respectively. The simulation results show that both approaches are able to solve all four problems. The results of the 20 region scenario are shown in Fig. 6. The path lengths of the proposed approach are comparable to TSP-CPP, being at worst 1.28% longer (20 regions) and at best 2.26% shorter (50 regions) than the path returned by TSP-CPP. It is worth noting that the inter-path length of the proposed approach is consistently longer than TSP-CPP for all trials, whereas the intra-path length is consistently shorter. This can be explained, as TSP-CPP considers the relationship between

inter and intra-region paths whereas our approach considers both problems separately. It can be seen that the region visiting order and inter-region paths differ between the solutions as TSP-CPP favors the inter-region paths at the cost of a longer intra-region path. It should be noted that the TSP-CPP solution often overshoots the boundaries of the regions. These overshoots can lead to collisions with obstacles and potential damage or loss of the robot.

To demonstrate the ability of the proposed approach to solve scenarios containing regions of any arbitrary shape, the approaches are compared on a scenario consisting of both convex and non-convex regions (see Fig. 7). Regions which fully enclose obstacles are excluded from the scenario since the TSP-CPP framework cannot model such polygons. However, as shown in Section 5.1, our proposed method successfully finds solutions for these scenarios as well. It can be seen that the TSP-CPP approach is unable to generate a coverage path for nonconvex regions (Fig. 7a). Even when convex decomposition is used to simplify the regions, the solution found by TSP-CPP connects subregions by crossing the polygon boundaries, resulting in an invalid path (Fig. 7b). On the other hand, our proposed approach is able to find a valid solution for this problem without crossing the region boundaries and only using the aerial modality to transition between regions where it is necessary (Fig. 7c). It should

Fig. 7 Comparison between the proposed method and the TSP-CPP method in a scenario with convex and non-convex regions



also be noted that while our approach is designed with 2.5D scenarios in mind, all scenarios in this Section have been constructed in 2D, as TSP-CPP is unable to process 2.5D scenarios.

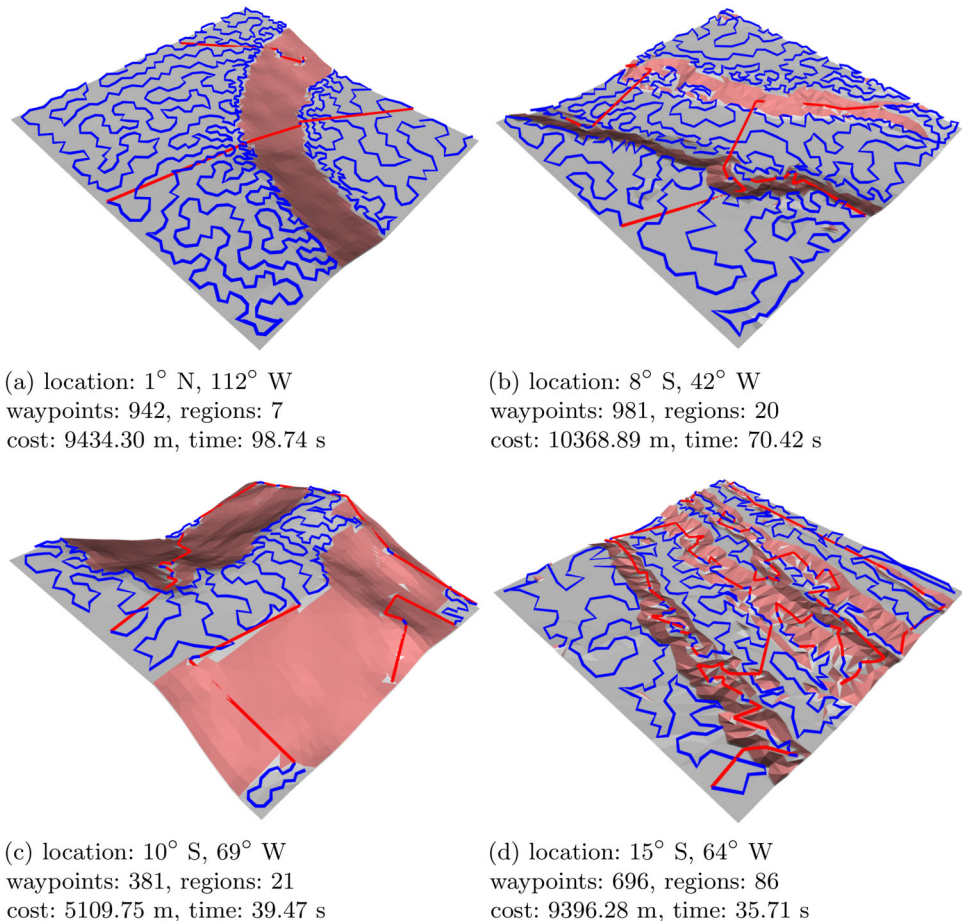
5.3 Scenarios with Martian Terrain Data

To evaluate the performance of the proposed method on realistic scenarios, four Martian terrain samples were collected from MarsTrek [63], which provides DEM data as STL files. The resulting maps have a size of 400×400 m and generally contain a similar amount of waypoints compared to the previous experiments. The results of this experiment are shown in Fig. 8. Figure 8a shows a map near a Martian crater that is solved by our approach, featuring two larger regions and a series of smaller regions along the crater edge. The approach can also solve valley scenarios where high slopes separate parts of the map (Fig. 8b) and hill scenarios where only narrow regions of the map can be covered safely (Fig. 8c). It can also solve highly complex scenarios containing a large number of waypoints and regions (Fig. 8d). All evaluated scenarios are successfully solved by the proposed approach.

The four scenarios have also been solved using the global TSP approach and the MILP based approach for comparison. The TSP-CPP approach was excluded from this experiment as it is unable to solve non-convex regions. While the proposed approach and the global TSP approach were running on a Raspberry Pi 4B, the MILP based approach was run on a personal computer due to compatibility issues as described in Section 5.2.2. Both approaches were able to solve all four scenarios and generated solutions with similar path lengths, but required significantly more time to find a solution. The paths generated by the global TSP approach are all slightly shorter than the proposed approach with improvements ranging from 0.50% to 4.34%. Paths generated by the MILP based approach are at most 2.28% shorter than those produced by the proposed method, with two of the four maps yielding paths of identical length.

In terms of calculation time, the proposed approach performs best, followed by the global TSP approach and finally the MILP based approach. The proposed approach solved the Martian scenarios 21.91% - 71.28% faster than the global TSP approach. Despite the MILP based approach running on more powerful hardware, the proposed approach solved

Fig. 8 Samples from the proposed algorithm run on MarsTrek DEM data [63]. Obstacles are shaded in light red on the terrain, gray regions are safe. Blue lines indicate the path taken inside of a region, while red lines indicate the use of the aerial modality



the scenarios 60.96% - 93.73% faster than the MILP based approach.

5.4 Scenarios with Large Regions

To determine the performance of the proposed approach in large complex scenarios, five large random maps were generated using the same approach as described above. Each map had a size of 60×60 meters, resulting in an area of 3600 m^2 with an average of 4019.00 nodes and 218.20 regions. Calculation time was limited to four hours per map.

The proposed approach was able to solve all five scenarios on the Raspberry Pi within an average of 24.08 minutes and a standard deviation of 14.09 minutes. The fastest solution was found within 137.41 seconds, and the slowest solution was found within 37.75 minutes. One example of the large scale maps is shown in Fig. 9.

For comparison, the same scenarios have been solved using the global TSP approach and the MILP based approach. While the proposed approach and the global TSP approach used the Raspberry Pi, the MILP based approach used a personal laptop due to compatibility issues as described above. The TSP-CPP approach was excluded from this experiment as it cannot solve non-convex regions.

The global TSP approach was able to solve all five scenarios with similar path lengths to the proposed method. In the worst case, the path generated by the proposed approach was 0.59% longer than the path generated by global TSP. In the best case, the path generated by the proposed approach was 5.44% shorter. Finding a solution with the the global TSP approach was between 3.60 and 37.87 times slower compared to the proposed approach.

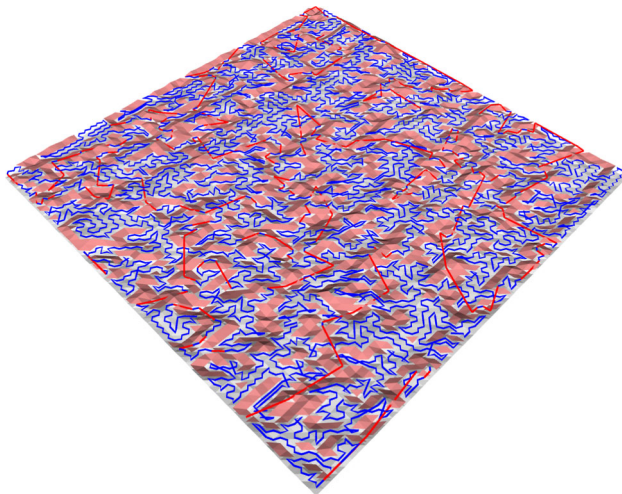


Fig. 9 Solution generated by the proposed approach on a large scenario containing 4271 waypoints in 185 disconnected regions. Obstacles are shared in light red on the terrain, gray regions are safe. Blue lines indicate the use of the ground modality, while red lines indicate the use of the aerial modality to transition between regions

Despite the stronger hardware, the MILP based approach was only able to solve one of the five scenarios within the given time frame. Three of the problems were terminated after exceeding four hours of calculation time and one problem was terminated due to running out of memory. The smallest of the five problems was solved successfully, yielding a path that is 0.04% shorter than the path generated by the proposed approach. The calculation time was 13.59 times slower than the proposed approach.

5.5 Complexity Analysis

The results gathered from the experiments offer insights into the space and time complexity of the proposed approach. The space complexity can be derived from the size of the distance matrices, which require the largest amount of memory as they need to store the distances between each pair of waypoints. In the global TSP scenario, this results in a space complexity of $\mathcal{O}(n^2)$, where n is the number of waypoints, which is equivalent to the worst case scenario for our proposed approach, where all waypoints belong to a single region. However, if the scenario consists of multiple disconnected regions, memory usage decreases, as distances between waypoints of different regions do not need to be calculated. In the best case, where regions are equally sized, the space complexity therefore reduces to $\mathcal{O}(\frac{n^2}{k} + k^2)$, where n is the number of waypoints and k is the number of regions.

Since the proposed method is heuristic, its time complexity cannot be derived analytically. Hence, a qualitative discussion is provided here for time complexity. Previous empirical studies have established the time complexity of the LKH algorithm for solving a global TSP as $\mathcal{O}(n^{2.2})$ [9]. The experimental results indicate that the proposed approach on average solves a problem significantly faster than the global TSP approach, suggesting a lower average time complexity. The calculation time appears to be positively correlated with the number of waypoints and the map size, and negatively correlated with the number of regions. However, it should be noted that a strong interdependency exists between these parameters, which might influence the computational complexity.

6 Conclusion

This work introduced a hierarchical TSP method for generating coverage paths for a multi-modal robot in 2.5D planetary exploration scenarios with multiple disconnected regions of arbitrary shapes. After identifying safe regions and obstacles from the DEM data of the target area, the approach generates the coverage path in two steps using the LKH-3 solver. First, a TSP is formulated and solved within each

individual disconnected region. Then, a second TSP is formulated and solved to connect all sub-paths and generate the final coverage path. Evaluation on 500 randomly generated maps with up to 1233 waypoints and up to 122 disconnected regions demonstrated that the method consistently generates coverage paths for complex scenarios with arbitrarily shaped regions and fully enclosed obstacles, while maintaining low computational requirements. Running on a Raspberry Pi 4B, the approach achieved an average computation time of 54.16 seconds, highlighting the method's low computational requirements and suitability for deployment in resource-constrained planetary robotics applications. The capabilities of the proposed approach were also demonstrated on realistic Martian surface data containing extreme terrain features such as craters and valleys, as well as large maps with up to 4344 waypoints.

Comparison with state-of-the-art CPP solvers further confirmed the efficiency and flexibility of the proposed approach. On average, the proposed approach finds a solution 40.08%

faster than a single tier global TSP formulation and 89.93% faster than a MILP based solver on small and medium sized maps. On larger maps, the performance difference becomes even more significant, with a speedup of 72.24% – 97.36% compared to the global TSP approach. At the same time, the path lengths generated by the proposed approach remain comparable to those found by the other methods, varying from 28.82% shorter to 4.84% longer. When compared to the TSP-CPP approach, the proposed approach demonstrates its ability to solve complex 2.5D scenarios with arbitrarily shaped disconnected regions.

Immediate future work considers path optimization with different routing approaches and dynamic constraints, as well as coverage in partially unknown environments.

Appendix A: Summary of Experimental Results

Table 1 Overview over all experimental results. The first three rows are aggregates over multiple samples. Values for these rows are given as averages with standard deviation($\pm$). The remaining rows are single samples and contain raw results. The Δ Cost and Δ Time columns contain a ratio relative to the proposed approach, and an average of ratios for

aggregate rows. (*) Calculation time cannot be compared as a different programming language was used. (†) Only one out of five samples was solved. (‡) Calculation times cannot be compared as a different device was used

Scenario	Nodes	Regions	Approach	Cost (m)	Δ Cost (%)	Time (s)	Δ Time (%)
15x15 Random	194.46	27.50	Proposed C++	171.56 $\pm$ 49.89	0.00	0.57 $\pm$ 0.94	~*
	$\pm$ 80.81	$\pm$ 11.82	Proposed Matlab	171.69 $\pm$ 49.90	0.08	3.93 $\pm$ 2.56	0.00
			MILP Based	171.65 $\pm$ 49.91	0.05	36.01 $\pm$ 83.09	442.75
25x25 Random	663.39	49.46	Proposed	527.85 $\pm$ 161.68	0.00	54.16 $\pm$ 64.97	0.00
	$\pm$ 302.42	$\pm$ 34.94	Global TSP	539.22 $\pm$ 144.57	3.82	68.06 $\pm$ 56.87	194.95
60x60 Random	4019.00	218.20	Proposed	3389.86 $\pm$ 432.12	0.00	1444.96 $\pm$ 845.28	0.00
	$\pm$ 598.10	$\pm$ 92.49	Global TSP	3419.65 $\pm$ 364.34	1.14	7983.60 $\pm$ 1658.11	107.32
			MILP Based	2617.82†	0.04†	1868.03†	1259.44†
Mars (a)	942	7	Proposed	9434.30	0.00	98.74	0.00
			Global TSP	9024.71	−4.34	186.57	88.94
Mars (b)	981	20	Proposed	10368.89	0.00	70.42	0.00
			Global TSP	10078.89	−2.81	147.19	109.02
Mars (c)	381	21	Proposed	5109.75	0.00	39.47	0.00
			Global TSP	4922.60	−3.66	50.54	28.07
Mars (d)	696	86	Proposed	9396.28	0.00	35.72	0.00
			Global TSP	9349.13	−0.50	124.34	248.12
TSP-CPP 2	27	2	Proposed	47.43	0.00	0.54	-
			TSP-CPP	48.28	1.78	0.11	~‡
TSP-CPP 20	2692	20	Proposed	4462.70	0.00	25.21	-
			TSP-CPP	4406.08	−1.27	0.20	~‡
TSP-CPP 50	1481	50	Proposed	2812.28	0.00	4.24	-
			TSP-CPP	2877.19	2.31	0.25	~‡
TSP-CPP 100	5661	100	Proposed	9952.62	0.00	24.07	-
			TSP-CPP	9846.91	−1.06	1.60	~‡

Author Contributions All authors contributed to the conceptualization and design. Methodology, software implementation, data collection and analysis were performed by S. S.. Supervision, project administration and additional resources were contributed by S.M., L. F. and S. U.. The first draft was written by S.S. and all authors commented on and edited previous versions of the manuscript. All authors read and approved the final manuscript.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data Availability The authors declare that the data supporting the findings of this study are available within the paper. Supplementary materials are available from the corresponding author on request.

Declarations

Competing Interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Tan, C.S., Mohd-Mokhtar, R., Arshad, M.R.: A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *IEEE Access*. **9**, 119310–119342 (2021)
2. Morrell, B., Thakker, R., Santamaria Navarro, À., Bouman, A., Lei, X., Edlund, J., Pailevanian, T., Vaquero, T.S., Chang, Y.L., Touma, T., et al.: Nebula: Team costar's robotic autonomy solution that won phase ii of darpa subterranean challenge. *Field Robot.* **2**, 1432–1506 (2022)
3. Green, J.J.K., Bozinovski, D., Tischhauser, F., Hutter, M., Baines, R.: Steerable high jumping tensegrity robot for space exploration. *IEEE Trans. Field Robot.* (2025)
4. Wang, F., Li, C., Niu, S., Wang, P., Wu, H., Li, B.: Design and analysis of a spherical robot with rolling and jumping modes for deep space exploration. *Machines*. **10**(2), 126 (2022)
5. Verma, V., Nash, J., Saldyt, L., Dwight, Q., Wang, H., Myint, S., Biesiadecki, J., Maimone, M., Tumber, A., Ansar, A., et al.: Enabling long & precise drives for the perseverance mars rover via onboard global localization. In: 2024 IEEE Aerospace Conference, pp. 1–18 (2024). IEEE
6. Xie, J., Carrillo, L.R.G., Jin, L.: Path planning for uav to cover multiple separated convex polygonal regions. *IEEE Access*. **8**, 51770–51785 (2020)
7. Stelter, S., U., D.K.I., Mondal, S., Felicetti, L., Upadhyay, S.: Hierarchical coverage path planning for a multi-modal robot exploring disconnected regions. In: Cavalcanti, A., Foster, S., Richardson, R. (eds.) *Towards Autonomous Robotic Systems*, pp. 433–444. Springer, Cham (2026)
8. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.* **21**(2), 498–516 (1973)
9. Helsgaun, K.: An effective implementation of the lin-kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* **126**(1), 106–130 (2000)
10. Honess, D., Quinlan, O.: Astro pi: Running your code aboard the international space station. *Acta Astronaut.* **138**, 43–52 (2017)
11. Drzadinski, N., Booth, S., LaFuente, B., Raible, D.: Space networking implementation for lunar operations. 37th Annual Small Satellite Conference. (2023)
12. Guertin, S.M.: Raspberry pi for space guideline. NASA: Washington, DC, USA. **6** (2022)
13. Araki, B., Strang, J., Pohorecky, S., Qiu, C., Naegeli, T., Rus, D.: Multi-robot path planning for a swarm of robots that can both fly and drive. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 5575–5582 (2017). IEEE
14. Mintchev, S., Zappetti, D., Willemin, J., Floreano, D.: A soft robot for random exploration of terrestrial environments. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 7492–7497 (2018). IEEE
15. Yang, Z., Zhu, S., Liang, Z., Xu, R.: Dual-stage control method for continuous hopping on the surface of small celestial bodies. *Acta Astronaut.* **222**, 436–449 (2024)
16. Jung, M., Chuen Tan, K., Dai, R.: Path planning for a jumping rover team with a charging station in multi-waypoints visiting missions. *Front. Control Eng.* **3**, 803468 (2022)
17. Tatar, A.B.: W-leg jumping robot: Mechanical design, dynamical analysis and simulation of jumping dual wheel-leg hybrid robot. *Arab. J. Sci. Eng.* **49**(11), 15463–15481 (2024)
18. Tanaka, K., Zhang, D., Inoue, S., Kasai, R., Yokoyama, H., Shindo, K., Matsuhira, K., Marumoto, S., Ishii, H., Takanishi, A.: A design of a small mobile robot with a hybrid locomotion mechanism of wheels and multi-rotors. In: 2017 IEEE International Conference on Mechatronics and Automation (ICMA), pp. 1503–1508 (2017). IEEE
19. Gefen, E., Zarrouk, D.: Flying star2, a hybrid flying driving robot with a clutch mechanism and energy optimization algorithm. *IEEE Access*. **10**, 115491–115502 (2022)
20. Anderson, J.L., Karras, J.T., Cacan, M., Kubiak, G., Pyrzak, G., Dor, H., Pipenberg, B.: Ingenuity, one year of flying on mars. In: 2023 IEEE Aerospace Conference, pp. 1–18 (2023). IEEE
21. Kikuchi, S., Watanabe, S.-I., Saiki, T., Yabuta, H., Sugita, S., Morota, T., Hirata, N., Hirata, N., Michikami, T., Honda, C., et al.: Hayabusa2 landing site selection: surface topography of ryugu and touchdown safety. *Space Sci. Rev.* **216**, 1–67 (2020)
22. Wang, R., Di, K., Wan, W., Liu, Z., Wang, Y., Liang, W., Chen, X., Zhi, S.: Topographic mapping and analysis based on 3d reconstruction model of simulated asteroid. *Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci.* **43**, 1165–1170 (2020)
23. Hedrick, G., Gu, Y.: Terrain-aware traverse planning for a mars sample return rover. *Adv. Robot.* **35**(21–22), 1286–1301 (2021)
24. Ono, M., Fuchs, T.J., Steffy, A., Maimone, M., Yen, J.: Risk-aware planetary rover operation: Autonomous terrain classification and path planning. In: 2015 IEEE Aerospace Conference, pp. 1–10 (2015). IEEE
25. Santra, S., Uno, K., Kudo, G., Yoshida, K.: Risk-aware coverage path planning for lunar micro-rovers leveraging global and local environmental data. *arXiv preprint arXiv:2404.18721*. (2024)
26. Souleiman, R.G., Varadharajan, V.S., Beltrame, G.: Multi-objective risk assessment framework for exploration planning using terrain and traversability analysis. *arXiv preprint arXiv:2410.03917*. (2024)
27. Azpúrua, H., Campos, M.F., Macharet, D.G.: Three-dimensional terrain aware autonomous exploration for subterranean and confined spaces. In: 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 2443–2449 (2021). IEEE

28. Nayak, S., Lim, G., Rossi, F., Otte, M., Croix, J.-P.: Multi-robot exploration for the cadre mission. *Auton. Robot.* **49**(2), 17 (2025)
29. Varadharajan, V.S., Beltrame, G.: A multi-robot exploration planner for space applications. *IEEE Robot. Autom. Lett.* (2025)
30. Akemoto, A., Zhu, F.: Informative path planning to explore and map unknown planetary surfaces with gaussian processes. In: 2022 IEEE Aerospace Conference (AERO), pp. 1–12 (2022). IEEE
31. Choset, H., Pignon, P.: Coverage path planning: The boustrophedon cellular decomposition. In: *Field and Service Robotics*, pp. 203–209 (1998). Springer
32. Zhang, P., Liu, Y., Du, H.: An integrated framework for uav-based precision plant protection in complex terrain: the achaga solution for multi-tea fields. *Front. Plant Sci.* **15**, 1440234 (2024)
33. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. *Ann. Math. Artif. Intell.* **31**(1), 77–98 (2001)
34. Tang, J., Mao, Z., Ma, H.: Large-scale multi-robot coverage path planning on grids with path deconfliction. *IEEE Trans. Robot.* (2025)
35. Nasirian, B., Mehrandeh, M., Janabi-Sharifi, F.: Efficient coverage path planning for mobile disinfecting robots using graph-based representation of environment. *Front. Robot. AI.* **8**, 624333 (2021)
36. Tunstel, E.: Mobile robotic surveying performance for planetary surface site characterization. In: *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pp. 200–205 (2008)
37. Tashtoush, T.H., Ruiz, C., Estevis, T., Herrera, E., Bernal, R., Martinez, R., Reyna, L.: Square spiral search (sss) algorithm for cooperative robots: Mars exploration. *Intern. J. Res.* **7**(1), 22–28 (2020)
38. Xie, Y., Hui, W., Zhou, D., Shi, H.: Three-dimensional coverage path planning for cooperative autonomous underwater vehicles: A swarm migration genetic algorithm approach. *J. Marine Sci. Eng.* **12**(8), 1366 (2024)
39. Khaloujini, A., Talaeizadeh, A., Taheri, A., Alasty, A.: Coverage path planning of agricultural spraying drones with graph convolutional network. In: 2024 12th RSI International Conference on Robotics and Mechatronics (ICRoM), pp. 543–548 (2024). IEEE
40. Hameed, I.A.: Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain. *J. Intell. Robot. Syst.* **74**(3), 965–983 (2014)
41. Qiu, W., Zhou, D., Hui, W., Kwabena, A.R., Xing, Y., Qian, Y., Li, Q., Pu, H., Xie, Y.: Terrain-shape-adaptive coverage path planning with traversability analysis. *J. Intell. Robot. Syst.* **110**(1), 41 (2024)
42. Helsgaun, K.: An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde Univ.* **12**, 966–980 (2017)
43. Modares, J., Ghanei, F., Mastronarde, N., Dantu, K.: Ub-anc planner: Energy efficient coverage path planning with multiple drones. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 6182–6189 (2017). IEEE
44. Bircher, A., Kamel, M., Alexis, K., Burri, M., Oettershagen, P., Omari, S., Mantel, T., Siegwart, R.: Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots. *Auton. Robot.* **40**(6), 1059–1078 (2016)
45. Reinelt, G.: Tspplib—a traveling salesman problem library. *ORSA J. Comput.* **3**(4), 376–384 (1991)
46. Pan, X., Jin, Y., Ding, Y., Feng, M., Zhao, L., Song, L., Bian, J.: H-tsp: Hierarchically solving the large-scale traveling salesman problem. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 9345–9353 (2023)
47. Yu, K., O’Kane, J.M., Tokekar, P.: Coverage of an environment using energy-constrained unmanned aerial vehicles. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 3259–3265 (2019). IEEE
48. Chen, X., Chen, J., Du, C., Xu, Y.: Region coverage path planning of multiple disconnected convex polygons based on simulated annealing algorithm. In: 2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET), pp. 238–242 (2021). IEEE
49. Chen, G., Shen, Y., Zhang, Y., Zhang, W., Wang, D., He, B.: 2d multi-area coverage path planning using l-shade in simulated ocean survey. **2021**, 112 (2021). <https://doi.org/10.1016/j.asoc.107754>
50. Du, L., Fan, Y., Gui, M., Zhao, D.: A multi-regional path-planning method for rescue uavs with priority constraints. *Drones.* **7**(12), 692 (2023)
51. Yu, X., Jin, S., Shi, D., Li, L., Kang, Y., Zou, J.: Balanced multi-region coverage path planning for unmanned aerial vehicles. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 3499–3506 (2020). IEEE
52. Lei, T., Luo, C., Jan, G.E., Bi, Z.: Deep learning-based complete coverage path planning with re-joint and obstacle fusion paradigm. *Front. Robot. AI.* **9**, 843816 (2022)
53. Majeed, A., Hwang, S.O.: A multi-objective coverage path planning algorithm for uavs to cover spatially distributed regions in urban environments. *Aerospace* **8**(11), 343 (2021)
54. Lyu, Y., Wang, S., Hu, T., Pan, Q.: Uav coverage path planning of multiple disconnected regions based on cooperative optimization algorithms. *IEEE Trans. Cogn. Develop. Syst.* (2024)
55. Ly, K.T., Munks, M., Merkt, W., Havoutis, I.: Asymptotically optimized multi-surface coverage path planning for loco-manipulation in inspection and monitoring. In: 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), pp. 1–7 (2023). IEEE
56. Plessen, M.: Path planning for spot spraying with uavs combining tsp and area coverages. *Smart Agricult Technol* 100965 (2025)
57. Wang, L., Zhuang, X., Zhang, W., Cheng, J., Zhang, T.: Coverage path planning for uavs: An energy-efficient method in convex and non-convex mixed regions. *Drones.* **8**(12), 776 (2024)
58. Smith, D.E., Zuber, M.T., Jackson, G.B., Cavanaugh, J.F., Neumann, G.A., Riris, H., Sun, X., Zellar, R.S., Coltharp, C., Connelly, J., et al.: The lunar orbiter laser altimeter investigation on the lunar reconnaissance orbiter mission. *Space Sci. Rev.* **150**(1), 209–241 (2010)
59. Cacciola, F., Rouxel-Labbé, M., Şenbaşlar, B.: Triangulated surface mesh simplification. *CGAL User Refer. Man.* **4**(1) (2016)
60. Lindstrom, P., Turk, G.: Fast and memory efficient polygonal simplification. In: *Proceedings Visualization’98* (Cat. No. 98CB36276), pp. 279–286 (1998). IEEE
61. Lawler, E.L.: The traveling salesman problem: a guided tour of combinatorial optimization. Wiley-Interscience Series in Discrete Mathematics. (1985)
62. Zomorodian, A., Edelsbrunner, H.: Fast software for box intersections. In: *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*, pp. 129–138 (2000)
63. Law, E., Day, B.: Mars trek: An interactive web portal for current and future missions to mars. In: *European Planetary Science Congress*, pp. 2017–99 (2017)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Sebastian Stelter received his M.Sc. degree in Computer Science from the University of Hamburg, Germany in 2022. He is currently a research student at Cranfield University. His research interests include mobile robotics, autonomous navigation and heterogeneous multi-agent systems.

Sabyasachi Mondal received the Ph.D. degree in Aerospace Engineering from the Indian Institute of Science, Bangalore, India, in 2018. He is currently a Lecturer at Cranfield University. His research interests include Guidance of Aerospace Vehicles, Cooperative control, Optimal control, Multi-Agent systems, and Path Planning.

Leonard Felicetti received the PhD degree from Sapienza University of Rome in 2023, where he also continued as a Postdoctoral Researcher. In 2015, he was an Honorary Research Associate at the University of Glasgow and then an Associate Senior Lecturer in On-Board Space Systems at Lulea University of Technology, Sweden. He joined Cranfield University in 2019 and is currently a Reader in Space Robotics and Guidance, Navigation and Control, and Course Director of the MSc in Astronautics and Space Engineering. Leonards leads both the teaching and research activities in guidance, navigation and control of space systems, spacecraft orbital and attitude control systems, and space robotics.

Saurabh Upadhyay received his B.E. degree from SSGMCE, Shegaon in 2009, M.Tech. degree from IIT Guwahati in 2012, and Ph.D. degree from IISc Bengaluru in 2018. He is a Lecture in Space Engineering at Cranfield University, UK. His research interests lie in mobile robots for extreme environments, with special focus on low onboard resources decision making and ISRU-enabled robot design.



DSPACE

<https://dspace.org/>

Computationally-fast coverage path planning for planetary multi-modal mobility robots traversing disconnected regions

Stelter, Sebastian; Mondal, Sabyasachi; Felicetti, Leonard; Upadhyay, Saurabh

2026-09

Springer

<https://doi.org/10.1007/s10846-026-02399-5>

Stelter S, Mondal S, Felicetti L, Upadhyay S. (2026) Computationally-fast coverage path planning for planetary multi-modal mobility robots traversing disconnected regions. Journal of Intelligent & Robotic Systems, Volume 112, Issue 3, September 2026, Article number 65