Minimising Revisits in Dubins-Constrained Grid Coverage

Sarah Frenkel[1], David Parker[2], and Masoumeh Mansouri[1]

***Abstract*— We study Dubins-constrained coverage on grids in applications where visiting a cell triggers an irreversible local action, such as painting or drilling. In these settings, re-entering treated cells destroys the initial action and is therefore undesirable, even if doing so would shorten the overall path. However, the motion constraints of a Dubins vehicle force entries into neighbouring cells during turns or when connecting distant regions. We refer to entries into previously treated cells as *revisits*. Motivated by this, we define the *Dubins Coverage with Minimal Revisits (DCMR)* problem: find a trajectory that covers every cell while minimising the number of revisits. We derive theoretical bounds on the number of revisits and boundary violations, and complement these results with a configurable solver supporting various decomposition and coverage pattern choices. We then experimentally evaluate how these choices affect the number of revisits and boundary violations.**

I. Introduction

We consider the problem of planning a coverage path for a robot operating in a known environment represented by an occupancy grid. The robot is modelled as a Dubins vehicle, that is, it moves forward only and has a fixed turning radius. Our goal is to analyse how different configurations of the coverage pipeline affect two quantities: *revisits*, where the trajectory re-enters cells that have already been covered, and *boundary violations*, where it enters non-free cells. Figure 1 illustrates both effects: Dubins-feasible turns may pass through previously covered cells (red) and may cross into non-free cells of the occupancy grid (blue).

The motivation for this model arises in applications where the robot performs a local action in each cell, such as painting, drilling, or surface inspection. After an action has been executed, revisiting the same cell is undesirable: it may damage the material or distort the result. At the same time, curvature constraints introduce a structural difficulty: during a turn, a Dubins vehicle necessarily enters neighbouring cells before reaching the next cell. The resulting trajectory may therefore revisit previously treated cells even if the discrete coverage order avoids repetition. Whether these additional entries lead to revisits or boundary violations depends on the geometry of the environment and the chosen coverage pattern. Since complete avoidance of such violations is typically impossible, the key question is not whether they occur, but how many are forced by the structure of the grid and the chosen coverage pattern.

The focus of this paper is how revisits and boundary violations arise from the structure of the environment and the configurations of the coverage pipeline. To study this systematically, we use a modular pipeline: (1) decompose the free space of an occupancy grid into rectangular regions (which we will refer to as rectangles) using different decomposition methods, (2) compute a visiting order of the rectangles, (3) choose a coverage pattern within each rectangle (spiral or back-and-forth), and (4) convert the discrete order into a Dubins-feasible trajectory by connecting consecutive rectangles with curvature-constrained trajectory segments. We refer to these additional segments as *connectors*. Based on the resulting trajectory, we count revisits to previously covered cells and entries into non-free cells.

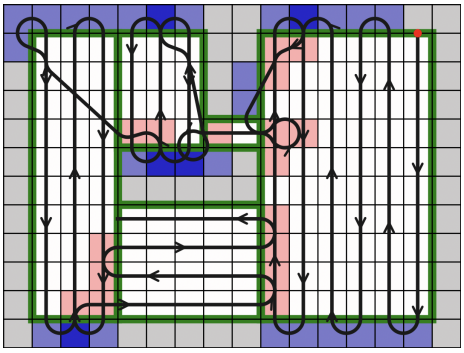


Fig. 1: Dubins-feasible coverage trajectory with revisit heatmap. Revisits (red) and boundary violations (blue) arise from turns and transitions between rectangles (green).

Contributions. We introduce the *Dubins Coverage with Minimal Revisits (DCMR)* as the problem of finding a trajectory that covers every cell while minimising the number of revisits. Unlike prior work on Dubins-constrained coverage, which focuses on solution feasibility or minimising path length, DCMR quantifies how curvature constraints induce repeated entries into cells in occupancy grids. For single rectangles, we prove bounds on the number of revisits and boundary violations induced by spiral and back-and-forth patterns. For general grid environments, we derive bounds for spiral patterns that depend only on the boundary of the environment, specifically on the number and type of its vertices. We then experimentally compare how different combinations of decomposition methods, ordering heuristics and coverage patterns affect revisits and boundary violations. Our experiments show that most revisits are induced by connectors between rectangles rather than by coverage patterns within a rectangle. Thus, the visiting order has a significantly greater influence on the number of revisits than the choice of decomposition or coverage pattern.

[1] School of Computer Science, University of Birmingham, United Kingdom, sxc1431@student.bham.ac.uk, m.mansouri@bham.ac.uk

[2] Department of Computer Science, University of Oxford, United Kingdom

II. Related work

Coverage path planning methods optimise trajectories that cover an environment under varying assumptions. Classical approaches assume static obstacles or obstacles whose dynamics are external to the robot [10], [5], [19]. The methods are evaluated by path length or energy consumption, without accounting for number of revisits under curvature constraints.

The initial step in addressing coverage path planning is decomposing the environment into several simpler shaped areas [6], [1]. In this work, we consider grid environments, which arise naturally when the motion pattern of a robot dictates a type of grid (in lattice-based planning [4]) or when the environment contains points of interest that are distributed in a grid-like fashion (in open-pit mining [12]). Decomposing such environments into rectangles is a classical problem with applications in areas such as VLSI design and image processing [18]. Once the environment is decomposed, the next task is to determine the visiting order in which the regions are visited. This can be formulated as a travelling salesman problem (TSP) or a generalised travelling salesman problem (GTSP). Recently, Bähnemann et al. [3] formulated boustrophedon coverage as a GTSP to optimise transitions between regions in static environments.

Dubins-constrained coverage has been studied in robotics, particularly for aerial or non-holonomic vehicles, where curvature constraints must be respected during coverage [16]. However, these works typically assume static environments and optimise path length or energy consumption. In contrast, we study the number of revisits and boundary violations induced by curvature constraints when covering grids.

The closest related model is due to Frenkel et al. [9], who study a Dubins-constrained robot on triangular grids in which visiting a vertex creates a circular obstacle. Their primary objective is to determine when complete coverage remains feasible under this traversal-induced obstruction. This paper also examines the same obstruction mechanism, but differs in several respects: we consider square occupancy grids, quantify revisits of previously covered cells and boundary violations induced by Dubins-feasible turns and connectors, and focus on comparing pipeline choices and deriving bounds rather than proving feasibility.

The existing literature treats grid decomposition, Dubins routing, and coverage objectives largely in isolation. Here, we analyse their interaction in a model where revisits are explicitly counted, and we derive bounds and comparative results for different decomposition and ordering strategies.

III. Problem description

We consider a *rectilinear* environment, i.e. a planar region with axis-aligned boundaries. We approximate the environment by an occupancy grid $\mathscr{G}$ of square cells of side length $2r$, where r is the turning radius of the robot. This guarantees feasibility of straight cell traversals under the Dubins constraints. This discretisation size may differ from the robot width and can be adjusted via the size of the end-effector. As in other grid-based planners, environments not perfectly divisible by the grid resolution can be handled through boundary padding. Our focus is on the structural impact of curvature constraints under a fixed discretisation.

The robot is modelled as a Dubins vehicle, i.e. a forward-only vehicle with fixed turning radius r. Motion between poses follows Dubins trajectories consisting of straight segments and circular arcs [7].

Our goal is to compute a continuous trajectory that visits all free cells while minimising revisits. To this end, we first determine a discrete ordering of the free cells, referred to as a *coverage sequence* $P = (p_1, ..., p_n)$. The coverage sequence P specifies the order in which cells are visited. Based on P, we construct a Dubins-feasible trajectory T connecting consecutive cells while respecting the turning constraint. Formally, T consists of trajectory segments $T = (t_1, ..., t_{n-1})$ where each segment t_i connects the traversal of cell p_i to the traversal of p_{i+1} while respecting the Dubins constraint. While P ensures every free cell is visited exactly once, T may incidentally traverse additional cells due to turning manoeuvres. In later sections, when rectangles are introduced, we refer to the trajectory segments connecting the traversal of different rectangles as connectors.

Revisits can occur for two reasons. First, the grid graph induced by the free cells of the environment may not permit a Hamiltonian path, so revisiting cell can be unavoidable even without the curvature constraints [2]. Second, even when such a path exists, the turning constraints of a Dubins vehicle may force the trajectory to enter additional cells while traveling between the cells in the coverage sequence. For the purposes of this paper, we do not distinguish between the causes and count all such entries as revisits. We formally define the problem as follows.

Problem 1: Given an occupancy grid $\mathscr{G}$ and the turning radius r of a Dubins-constrained robot, where each cell of the grid $\mathscr{G}$ has side length $2r$, the *Dubins Coverage with Minimal Revisits* problem $\mathrm{DCMR}(\mathscr{G}, r)$ is to find a trajectory T based on a coverage sequence $P = (p_1, ..., p_n)$ through the free cells of $\mathscr{G}$, such that:

- every free cell is visited exactly once by P,
- a visit to a cell implies that the trajectory traverses the cell in a straight line, entering at the midpoint of one side and exiting at the midpoint of the opposite side.
- T is a feasible Dubins trajectory consisting of segments $T = (t_1, ..., t_{n-1})$, where each segment t_i connects p_i to p_{i+1}, respects the Dubins turning constraint, and ends at the same pose from which the next segment begins,
- and the number of revisits

$$R(T) = \sum_{i=1}^{n} \left| C_i \cap P_{i-1} \right|,$$

is minimised, where C_i is the set of cells entered by the trajectory segment t_i, and $P_{i-1} = \{p_1, \ldots, p_{i-1}\}$ is the set of previously visited cells.

Besides revisits, we measure boundary violations. Let O denote the set of non-free cells in $\mathscr{G}$. The number of boundary violations of a trajectory T is defined as $B(T) = \sum_{i=1}^{n} |C_i \cap O|$, that is, the number of trajectory entries into non-free cells.

The DCMR generalises classical coverage path planning problems by introducing Dubins constraints and a revisit metric. Even without curvature constraints, determining an optimal order in which to visit all free cells corresponds to the Hamiltonian path problem on the grid graph induced by the free cells of the occupancy grid. Since the Hamiltonian path problem on grids is NP-hard [2] in the general case, the DCMR is NP-Hard as well.

IV. Theoretical Problem Analysis

In this section, we prove structural bounds on revisits and boundary violations based on two different coverage patterns.

The *spiral pattern* traverses a rectangle from a central cell moving outwards in concentric layers, as seen in Fig. 2a. The *back-and-forth pattern* sweeps an area in alternating directions, either in rows or columns, as seen in Fig. 2b.

Lemma 1: Let $(\mathscr{G}, r)$ be an instance of the DCMR, where $\mathscr{G}$ is an $n \times m$ grid, with $n \leq m$. If the rectangle is covered with a trajectory based on a *spiral pattern P*, then

- the number of revisits is 0,
- the total number of boundary violations is 11.

Proof: A trajectory based on a spiral pattern on $\mathscr{G}$ has $2(n-1)$ turns. Under Dubins constraints, each turn enters the planned cell and three adjacent cells. Since the spiral expands outward, these adjacent cells have not been deliberately visited yet and will be covered later. Hence, no revisit occurs. Near the boundary, four turns occur (Figure 2a). At the first boundary turn, two of the three additional cells lie outside the boundary of the rectangle. At each of the next three turns, all three additional cells are outside the rectangle. Thus the total number of boundary violations is $2+3\cdot 3 = 11$. ■

Lemma 2: Let $(\mathscr{G}, r)$ be an instance of the DCMR, where $\mathscr{G}$ is an $n \times m$ grid, with $n \leq m$. If the rectangle is covered with a trajectory based on a *back-and-forth pattern P*, then

- the number of revisits is 0,
- the total number of boundary violations is $2n-2$.

Proof: Since each U-turn is executed entirely outside the rectangle (Fig. 2b), no revisits occur. The $n-1$ U-turns each enter two cells outside the rectangle, resulting in $2(n-1)$ boundary violations. ■

We extend these bounds to general grid environments using structural properties of rectangular decompositions. We use the following classical bound on minimal rectangular decompositions (see Fig. 3).

Theorem 3 ([11], [14], [8]): An orthogonal polygon can be minimally partitioned into $c_n - l - h + 1$ rectangles where c_n is the number of non-convex vertices h is the number of holes and l is the maximum number of non-intersecting chords that can be drawn either horizontally or vertically between reflex vertices.

By Theorem 3, the number of rectangles in a minimal decomposition depends only on c_n, l, and h. Therefore, the number of turns created by rectangle boundaries is also controlled by these parameters, which allows us to bound the total number of violations in terms of c_n and l.

Lemma 4: Let $(\mathscr{G}, r)$ be an instance of the DCMR, let c_c, c_n be the number of convex and non-convex vertices, and l the maximum number of non-intersecting chords of $\mathscr{G}$.

Using the spiral pattern,

- the total number of revisits is at most $7c_n - 10l$,
- and the total number of boundary violations is at most $3c_n + c_c$.

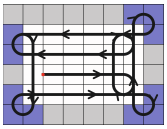
Proof: By Lemma 1, additional entries into cells occur only at vertices of the decomposition. We classify vertices into three types: (i) convex vertices, (ii) non-convex vertices incident to a non-intersecting chord and (iii) remaining non-convex vertices. Since each chord has two endpoints, there are exactly $2l$ vertices of type (ii) and $c_n - 2l$ of type (iii).

A convex corner induces at most 3 boundary violations and no revisits. Each non-intersecting chord endpoint induces at most 2 boundary violations and 4 revisits.

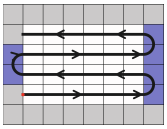
For a non-convex vertex not incident to a non-intersecting chord, an internal split-vertex appears between three adjacent rectangles, for example between the two blue and the green rectangle in Fig. 4c. In the worst case the adjacent rectangles are covered before the incident turn is executed. In this configuration, the turn may re-enter previously covered cells, contributing at most 7 revisits and 1 boundary violation.

Summing over all vertices yields $4l + 7(c_n - 2l) = 7c_n - 10l$ revisits, and $3c_c + 2l + (c_n - 2l) = 3c_c + c_n$ boundary violations. ■

The bounds on revisits using the spiral pattern on a rectilinear environment depend only on the boundary structure of the environment, additional entries into cells happen at concave and convex corners. Therefore, we can express the number of revisits and boundary violation in terms of the number of reflex vertices and non-intersecting chords. In contrast, for a back-and-forth pattern, the reason for boundary violations is the repeated U-turn at the end of each line. The number of such turns depends on pattern direction and rectangle size and not on the number and type of vertices

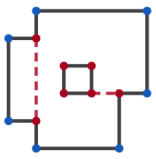


(a) Spiral pattern

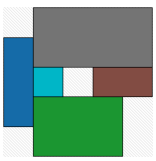


(b) Back-and-forth pattern

Fig. 2: Trajectories based on different coverage patterns, starting from the red dots, blue cells are boundary violations.



(a) An orthogonal polygon with one hole, 7 convex vertices (blue), 7 non-convex vertices (red), and 2 non-intersecting chords (red).



(b) A minimal decomposition of the polygon into five rectangles.

Fig. 3: Illustration of the rectangle decomposition bound from Theorem 3.

(a) Greedy (b) Largest empty rectangle

(c) Graph based (d) Boustrophedon

Fig. 4: Rectangle decomposition using different approaches.

of the environment. Thus, no bound analogous to Lemma 4 can be expressed purely in terms of (c_c, c_n, l).

The previous bounds quantify the violations that arise from the coverage patterns *within* rectangles. However, rectangle traversals must be connected by Dubins-feasible connectors. These connectors are often the main source of revisits in practice: even if each rectangle is covered with zero revisits, the Dubins-feasible connector between rectangles may re-enter previously covered areas. The experimental results in Section VI capture both the effect of the chosen coverage pattern and the effect of the visiting order.

V. Method

We now describe the modular coverage pipeline used to solve Problem 1. The pipeline consists of four stages:

1) Decompose the free area of the environment into several non-overlapping rectangles.
2) Decide on a visiting order of the rectangles.
3) Based on the visiting order and the coverage pattern, generate the coverage sequence of the cells in each rectangle.
4) Join the individual sequences into one coverage sequence and generate a Dubins trajectory.

A. Decomposition methods

We compare four decomposition methods for partitioning the free space into non-overlapping rectangles. Fig. 4 shows the output of the methods applied to the same input grid.

1) Greedy approach: The first approach greedily decomposes the free space into rectangles. The grid is scanned from top to bottom and left to right. Whenever an unvisited free cell is encountered, the algorithm grows the largest rectangle by first extending horizontally and then vertically while maintaining the same width. The rectangle is recorded and its cells are marked as covered. This process continues until the grid is fully covered (see Fig. 4a). The method is fast but results in more rectangles than the other methods.

2) Largest empty rectangle: We compute all maximal empty rectangles and iteratively select the one covering the largest number of uncovered free cells, marking cells as covered after each step. When no further rectangles fit, the remaining cells are covered using the greedy method from Section V-A.1, as seen in Fig. 4b.

3) Graph based decomposition: Several authors independently showed that a decomposition using the minimum number of rectangles can be obtained by selecting a set of non-overlapping cuts between concave corners of the shape [11], [14], [8]. Following this idea, we first compute all admissible cuts between concave vertices and select a largest non-crossing subset. The resulting regions are then subdivided into rectangles by adding horizontal lines through the remaining concave corners (Fig. 4c).

4) Boustrophedon: The Boustrophedon method is a classical cellular decomposition approach [6]. Depending on the sweep direction it produces horizontal or vertical strips. We use a vertical sweep, producing strips that extend maximally on the occupancy grid.

B. Determine visiting order

After decomposing the environment into rectangles, the next step is to compute a visiting order of the rectangles.

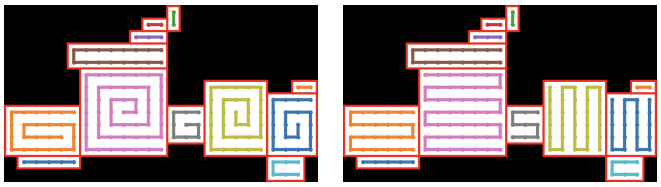
1) Centre-based TSP heuristic: As a baseline visiting-order heuristic, we construct a TSP, where the nodes are the centers of the rectangles, and the distances are given by their distance in the free space of the grid using A*. Solving this TSP creates an initial order. If a rectangle were covered immediately when first encountered, later transitions between the rectangles could enter it again and thus induce revisits. To avoid this, we construct the expanded order by recording, for each pair of consecutive rectangles on the initial order, the shortest path between them in the adjacency graph of the rectangles. Such a path may pass through rectangles that appear earlier in the initial order. The final visiting order is obtained from the expanded order by keeping only the last occurrence of each rectangle. This ensures that each rectangle is covered only after its last traversal by any connector induced by the visiting order.

2) GTSP-based ordering with entry-exit modelling: We also use a generalised travelling salesman (GTSP) formulation that reasons over visiting order and entry-exit configurations jointly. For each rectangle R_i, we pre-compute all possible entry–exit pose pairs and treat each pair as a node. Nodes corresponding to the same rectangle form one cluster.

For two configurations u and v belonging to different rectangles R_i and R_j, we define a directed transition cost

$$c(u,v) = d_{\mathrm{A}^*}(u_{\mathrm{exit}}, v_{\mathrm{entry}}) + \lambda h(R_i, R_j).$$

Here, $d_{\mathrm{A}^*}(u_{\mathrm{exit}}, v_{\mathrm{entry}})$ denotes the A* distance in the free space of the grid between the exit pose u_{exit} and the entry pose v_{entry}. The term $h(R_i, R_j)$ denotes the shortest-path distance between R_i and R_j in the rectangle adjacency graph. To balance geometric and topological terms we introduce the scaling factor $\lambda = {}^{\mathrm{median}(d_{A*})}/_{\mathrm{median}(h)}$, computed over all transitions between rectangles. The hop-penalty term $\lambda h(R_i, R_j)$ discourages transitions between rectangles that are far apart in the decomposition, even if their entry and exit poses are close. Without this term, the solver may prefer



(a) Spiral pattern. (b) Back-and-forth pattern.

Fig. 5: Individual coverage sequences for each rectangle.

geometrically short but topologically disruptive jumps that later induce additional revisits.

The GTSP instance is reduced to an Asymmetric TSP (ATSP) using the Noon–Bean transformation [13], which we solve with OR-Tools [15]. We use parallel cheapest insertion to generate an initial tour, followed by guided local search for improvement. This generates a tour that encodes both a rectangle order and entry-exit pair per rectangle.

Directly optimising revisits within the ordering stage is computationally expensive. Evaluating an ordering candidate requires the creation of the Dubins trajectory and the creation of the heatmap. Embedding this simulation that would need to be recalculated for each new candidate would increase computational costs substantially. We therefore use geometric (d_{A*}) and topological distances (h) as proxies.

C. Generate coverage sequences for individual rectangles

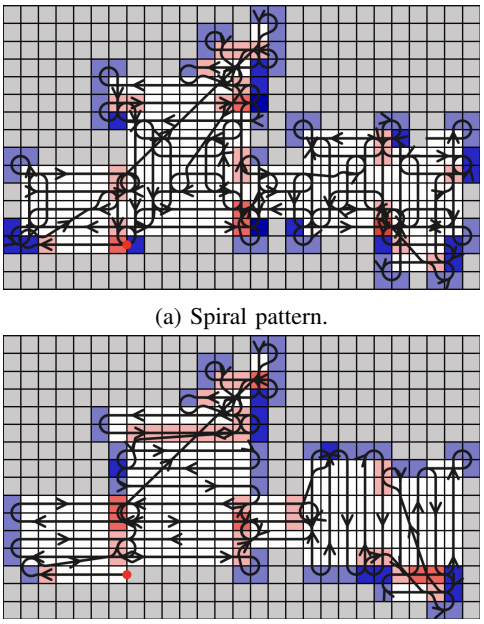
If the visiting order is obtained from the centre-based TSP heuristic, the exit point of each rectangle is chosen greedily as the corner closest to the centre of the next rectangle in the order. If a GTSP-based ordering is used, the exit pose is taken from the GTSP solution.

The spiral generation starts at the exit point and moves along the rectangle boundary, proceeding inwards in layers towards the center. The resulting sequence is then reversed. Figure 5a shows an example output. The back-and-forth pattern generates a lawnmower-style coverage sequence through a grid, as shown in Fig. 5b. The traversal alternates directions on each row or column, forming a back-and-forth pattern.

D. Join sequences and compute Dubins trajectory

With the results from Sections V-B and V-C, we obtain an ordered collection of coverage sequences of the rectangles together with the corresponding entry and exit poses. These sequences are joined in order to form a single coverage sequence. The final trajectory is obtained by connecting consecutive poses with shortest Dubins trajectories.

Within a rectangle, consecutive poses correspond to neighbouring cells and are connected by shortest Dubins trajectories. When transitioning from the exit of one rectangle to the entry of the next, a connector is computed. Connectors are computed by planning an A* path in the grid and converting the resulting polyline into a Dubins-feasible trajectory via waypoint smoothing. The connector consists of a sequence of Dubins trajectories between successive waypoints. Example trajectories generated based on spiral and back-and-forth patterns are shown in Fig. 6.



(a) Spiral pattern.

(b) Back-and-forth pattern.

Fig. 6: The calculated Dubins trajectories and heatmaps, counting revisits (red) and boundary violations (blue).

VI. Experiments

Our experiments compare how different pipeline configurations affect revisit numbers under Dubins constraints. We vary the decomposition method, coverage pattern, and visiting order, while keeping all other components fixed.

A. Setup and metrics

We evaluate our approach on occupancy grids from the MovingAI benchmark dataset [17]. For each map, we solve the DCMR using the pipeline in Section V and compare configurations independently. We measure the number of revisits and number of revisited cells, boundary violations, and trajectory length. The number of revisits counts how often the trajectory enters a cell after it has already been visited. In contrast, the number of revisited cells counts how many distinct cells are entered again after their visit. Boundary violations count entries into non-free cells. We compute these metrics by rasterising each trajectory and record how often it enters grid cells.

B. Results

Table I compares pipeline configurations. Each row corresponds to a triple (decomposition, pattern, ordering), with metrics reported as medians across all maps. The centre-based TSP achieves fewer revisits than the GTSP ordering and reduces boundary violations and trajectory length. This suggests that the GTSP cost function does not fully capture whether connectors pass through already covered cells.

Table II reports paired differences when varying pipeline configuration. The largest improvement occurs when replacing GTSP with TSP ordering, while coverage pattern and decomposition have smaller effects. The reduction of violations when using the graph-based decomposition is consistent with its fewer rectangles and hence fewer connectors.

decomposition	pattern	ordering	revisits	boundary	length	revisited cells	time (s)
greedy	back-and-forth	TSP	96	121	1056.59	86	5.12
boustrophedon	spiral	TSP	97	130	1234.42	91	5.12
largest	spiral	TSP	98	125	1270.15	71	5.12
largest	back-and-forth	TSP	99	121	1037.14	76	5.12
boustrophedon	back-and-forth	TSP	104	130	1162.24	101	5.12
graph	back-and-forth	TSP	105	115	1023.32	92	5.11
greedy	spiral	TSP	107	118	1214.12	89	5.12
graph	spiral	TSP	114	129	1171.87	92	5.12
greedy	spiral	GTSP	128	129	1138.8	109	21.74
graph	spiral	GTSP	130	128	1105.82	105	17.62
boustrophedon	spiral	GTSP	139	129	1089.22	122	18.63
boustrophedon	back-and-forth	GTSP	150	135	1008.5	134	17.3
graph	back-and-forth	GTSP	163	129	1011.89	143	17.65
largest	spiral	GTSP	170	128	1236.88	135	18.7
greedy	back-and-forth	GTSP	180	142	1039.79	147	19.84
largest	back-and-forth	GTSP	201	141	1059.42	166	18.72

TABLE I: Median performance of all pipeline configurations with TSP and GTSP ordering. Pipelines are ranked by revisits; metrics are aggregated over all benchmark maps.

Comparison	revisits	boundary	length
ordering: gtsp → tsp	-59	-3	-2.33
pattern: spiral → back-and-forth	6	-8	-162
decomposition: greedy → graph based	-9	-5	-15.2
decomposition: greedy → largest	0	0	0.6
decomposition: greedy → boustrophedon	-3	7	-10.3

TABLE II: Median paired differences between pipeline components.

VII. Discussion and Conclusion

We analysed how decomposition methods, ordering, and coverage patterns affect revisits under Dubins constraints. In rectangular regions, spiral and back-and-forth patterns exhibit distinct structural behaviours. Spiral patterns induce a constant number of boundary violations, whereas the back-and-forth pattern scales with the size of the rectangle. For general environments, we derive bounds for spiral patterns that depend only on the environment's boundary, specifically on the number and types of its vertices.

Experimentally, ordering has the highest impact on revisit numbers. Replacing GTSP ordering by a centre-based TSP heuristic reduces revisits, with only marginal effects on trajectory length. The coverage pattern has a secondary influence and the decomposition method the smallest effect among the tested components. Overall, the results indicate that the number of revisits in Dubins-constrained grid coverage is mainly determined by the geometry of the connectors. By choosing an ordering where these connectors lie in the not yet covered space, revisits can be reduced.

In future work, we will investigate alternative models for connector cost that better account for revisits while avoiding explicit trajectory evaluation. Additionally, we will explore the impact of robot footprints in the decomposition and their influence on the overall pipeline.

References

[1] Ercan U Acar, Howie Choset, Alfred A Rizzi, Prasad N Atkar, and Douglas Hull. Morse decompositions for coverage tasks. *The international journal of robotics research*, 21(4):331–344, 2002.

[2] Esther M Arkin, Sándor P Fekete, Kamrul Islam, Henk Meijer, Joseph SB Mitchell, Yurai Núñez-Rodríguez, Valentin Polishchuk, David Rappaport, and Henry Xiao. Not being (super) thin or solid is hard: A study of grid Hamiltonicity. *Computational Geometry*, 42(6-7):582–605, 2009.

[3] Rik Bähnemann, Nicholas Lawrance, Jen Jen Chung, Michael Pantic, Roland Siegwart, and Juan Nieto. Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem. In *Field and Service Robotics: Results of the 12th International Conference*, pages 277–290. Springer, 2021.

[4] Alexander Botros and Stephen L Smith. Computing a minimal set of t-spanning motion primitives for lattice planners. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2328–2335. IEEE, 2019.

[5] Tauã M Cabreira, Lisane B Brisolara, and Ferreira Jr Paulo R. Survey on coverage path planning with unmanned aerial vehicles. *Drones*, 3(1):4, 2019.

[6] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics*, pages 203–209. Springer, 1998.

[7] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[8] L Ferrari, Pathamadi V Sankar, and Jack Sklansky. Minimal rectangular partitions of digitized blobs. *Computer vision, graphics, and image processing*, 28(1):58–71, 1984.

[9] Sarah Frenkel, David Parker, and Masoumeh Mansouri. Coverage with self-induced obstacles on grids. *IEEE Robotics and Automation Letters*, 2026.

[10] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous systems*, 61(12):1258–1276, 2013.

[11] Witold Lipski Jr, Elena Lodi, Fabrizio Luccio, Cristina Mugnai, and Linda Pagli. On two-dimensional data organization ii. *Fundamenta Informaticae*, 2(1):245–260, 1979.

[12] Masoumeh Mansouri, Fabien Lagriffoul, and Federico Pecora. Multi vehicle routing with nonholonomic constraints and dense dynamic obstacles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3522–3529. IEEE, 2017.

[13] Charles E Noon and James C Bean. An efficient transformation of the generalized traveling salesman problem. *INFOR: Information Systems and Operational Research*, 31(1):39–44, 1993.

[14] Tatsuo Ohtsuki. Minimum dissection of rectilinear regions. In *Proc. 1982 IEEE Symp. on Circuits and Systems, Rome*, pages 1210–1213, 1982.

[15] Laurent Perron and Vincent Furnon. Or-tools.

[16] Ketan Savla, Emilio Frazzoli, and Francesco Bullo. Traveling salesperson problems for the dubins vehicle. *IEEE Transactions on Automatic Control*, 53(6):1378–1391, 2008.

[17] N. Sturtevant. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2):144 – 148, 2012.

[18] Tomáš Suk, Cyril Höschl IV, and Jan Flusser. Decomposition of binary images—a survey and comparison. *Pattern Recognition*, 45(12):4279–4291, 2012.

[19] Chee Sheng Tan, Rosmiwati Mohd-Mokhtar, and Mohd Rizal Arshad. A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *Ieee Access*, 9:119310–119342, 2021.