MDPI

Article

Exact and Heuristic Multi-robot Dubins Coverage Path Planning for known environments

Lin Li[1], Dianxi Shi [2,3]*, Songchang Jin [2,3]*, Shaowu Yang [1]*, Chenlei Zhou [3], Yaoning Lian[1], and Hengzhu Liu[1]

[1] College of Computer, National University of Defense Technology, Changsha, China.
[2] Artificial Intelligence Research Center (AIRC) Defense Innovation Institute, Beijing, China.
[3] Tianjin Artificial Intelligence Innovation Center (TAIIC), Tianjin, China.
* Correspondence: dianxishi@nudt.edu.cn, jsc04@tsinghua.org.cn

Abstract: Coverage path planning (CPP) of multiple Dubins robots has been extensively applied in aerial monitoring, marine exploration, and search and rescue. Existing multi-robot coverage path planning (MCPP) works use exact or heuristic algorithms to address coverage applications. However, several exact algorithms always provide precise area division rather than coverage paths, and heuristic methods face the challenge of balancing accuracy and complexity. This paper focuses on the Dubins MCPP problem of known environments~~for multiple Dubins robots~~. Firstly, we present an exact Dubins MCPP (EDM)~~EMD~~ algorithm based on~~modelling the MCPP problem as~~ mixed linear integer programming (MILP). The EDM algorithm searches the entire solution space to obtain the shortest Dubins coverage path.~~provides precise coverage paths by simultaneously optimizing task allocation and path planning subproblems.~~ Secondly, a heuristic approximate credit-based Dubins MCPP (CDM)~~CMD~~ algorithm is presented, which utilizes the credit model to balance tasks among robots and a tree partition strategy to reduce complexity. Comparison experiments with other exact and approximate algorithms demonstrate that EDM~~EMD~~ provides the least coverage time in small scenes, and CDM~~CMD~~ produces shorter coverage time and less computation time in large scenes. Feasibility experiments demonstrate the applicability of EDM~~EMD~~ and CDM~~CMD~~ to a high-fidelity fixed-wing unmanned aerial vehicle (UAV) model.

Keywords: complete coverage, Dubins robot, path planning

0. Introduction

As one subproblem of robot path planning, coverage path planning (CPP) aims to determine~~is the task of determining~~ a path to visit all points in the region [1]. CPP is common to several applications, including~~Various robotic applications requiring CPP include~~ small-scale household tasks like floor cleaning or lawn mowing and large-scale operations like search and rescue and environmental monitoring [2]. Due to the limited sensing range, calculating speed, and energy supply, many practical coverage applications cannot be achieved by a single robot [3]. Thus, a series of multi-robot CPP (MCPP) algorithms have been proposed to improve coverage efficiency and enhance robustness. Meanwhile, MCPP faces the challenges of algorithmic complexity and logistical management [39]. ~~The CPP problem's complexity depends on factors such as the complexity of the environments, and the kinematic constraints of robots [4]. In general, a robot with nonholonomic constraints is more challenging to plan than one with holonomic constraints [5], and an obstacle-constrained environment is more involved in planning than one without obstacles [6].~~

Real-world MCPP applications, such as aerial monitoring [7], marine exploration[8], and automatic farming [9][10], typically involve multiple aerial (fixed-wing aircraft), ground (wheel robots), and autonomous underwater/surface vehicles. These vehicles are typically governed by the Dubins vehicle model [32], which allows them to move at a fixed speed and turn with a limited turning radius. As the foundation of many practical

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

applications, MCPP oriented to Dubins robots (Dubins MCPP) has received growing attention in recent years. Thus, this paper focuses on the Dubins MCPP problem of known environments.

~~Utilizing multiple robots can reduce coverage time and increase robustness [11]. However, multi-robot CPP (~~ MCPP problem has been proven to be NP-hard [12]. Various MCPP works have been proposed to address the MCPP problem, and the related reviews can be found in [13][14][15]. Existing MCPP methods can be classified as exact or heuristic according to their accuracy [16]. Exact methods can provide the optimal solution for small-scale coverage applications. Heuristic methods are used to get a (near-) optimal result for large-scale coverage applications since they often involve a great number of tasks.

Existing MCPP methods perform well, but they suffer from three issues. The first issue is that several exact methods always provide an accurate partition of the region. However, the accurate partition is not equivalent to optimal coverage paths for the MCPP problem. Second, heuristic methods always face the challenge of how to balance accuracy and complexity. Traditional heuristic methods represent coverage tasks as graphs and obtain an efficient result using graph-partition and tree-partition strategies. In the graph-partition strategy, all vertexes and edges are considered to achieve near-optimal results. However, its runtime increases since the search space increase exponentially with the number of vertexes [16]. The tree-partition strategy compresses the search space of the MCPP problem by pruning the edges of the graph. While the compressed search space reduces the runtime, it decreases the accuracy of the solution. The third issue is that many MCPP works have been proposed, but only a few studies have been conducted on the Dubins robot. As a curvature-constrained robot, the Dubins robot cannot recede and can only move at a fixed speed and with a bound curvature. Without consideration of robot kinematics, MCPP algorithms probably generate piecewise paths that are only comprised of straight lines and sharp turns [17]. However, these paths are not feasible to follow for Dubins robots.

This paper presents two algorithms to address the Dubins MCPP~~multi-robot Dubins CPP (MDCPP)~~ problem. First, an exact Dubins MCPP (EDM)~~multi-robot Dubins CPP (EMD~~) algorithm is proposed, formulating the Dubins MCPP problem as a MILP to produce the optimal Dubins coverage paths. Second, we present a heuristic approximate credit-based ~~multi-robot~~ Dubins MCPP (CDM)~~CMD~~ algorithm. CDM~~CMD~~ divides the region into multiple partitions by a tree-partition strategy and balances coverage tasks among partitions by the credit model [18]. The effectiveness of EDM~~EMD~~ and CDM~~CMD~~ was validated in comparison and feasibility experiments. In summary, the contributions of this paper are as follows:

- We present an EDM~~EMD~~ algorithm based on MILP, which provides the shortest Dubins coverage path by searching the entire solution space. ~~modeling the MCPP problem as MILP.~~
- We present a CDM~~CMD~~ algorithm, which ensures the task balance among robots by the credit model and reduces complexity by a tree-partition strategy.
- Extensive validations. (i) Comparison experiments with other exact and heuristic MCPP methods show that EDM~~EMD~~ provides the minimum coverage time in small coverage scenes, and CDM~~CMD~~ generates a shorter coverage time and less computation time in large coverage scenes. (ii) Feasibility experiments are conducted on a high-fidelity UAV model to validate the applicability of EDM~~EMD~~ and CDM~~CMD~~.

The remaining of this paper is organized as follows. In Section 1, the related works are reviewed. Section 2 states the Dubins MCPP~~DMCPP~~ problem and presents a Dubins coverage framework. Section 3 and 4 describe EDM~~EMD~~ and CDM~~CMD~~, including their ideas and implementation. The comparison and feasibility tests of EDM~~EMD~~ and CDM ~~CMD~~ are presented in Section 5, followed by the paper's conclusion.

1. Related Work

1.1. Exact and heuristic MCPP methods

Existing works use exact and heuristic algorithms to solve the MCPP problem. Exact algorithms guarantee an optimum solution, while heuristic algorithms seek to yield a good, but not necessarily optimal, solution. However, an exact algorithm takes much longer to find an optimum solution than a heuristic one to a difficult problem [19]. Thus, exact algorithms are suitable for small-scale applications, whereas large-scale coverage applications often use heuristics to achieve a suboptimal solution.

Exact MCPP algorithms use the MILP [20][21], branch and bound method [22], and dynamic programming method to obtain the optimum solution. Some exact algorithms precisely divide the region into K partitions and apply a single-robot coverage algorithm to each partition. For example, the work [23] proves that BCP_q is NP-hard and proposes a polynomial-time algorithm that solves BCP_2 within an approximation ratio of $\frac{4}{3}$. [21] proposes an accurate MILP for two robots, which can solve graphs of 70 nodes within two hours. However, the MILP is difficult to extend to cases with $q \geq 3$. The work [20] transforms the MCPP problem into the $MinMax$ balanced and connected q-partition problem (BCP_q) and presents an exact Milpflow algorithm to handle it. The Milpflow algorithm provides a precise partition of the region through a flow model and applies a single-robot CPP algorithm to each partition. However, the optimial partition is not equivalent to optimal coverage paths. Other exact works build exact formations based on MILP to generate the shortest or fastest paths. For example, the works [3][16] produces the fastest coverage paths by building exact formulations. However, both works calculate the coverage time of a given region based on the scanning area rather than the coverage path. In fact, the coverage time of a given region depends on the time to cover the scanning area and the time to perform turns. Since turns are often costly for mobile robots, neglecting the cost of turns usually reduces the efficiency [24]. In order to minimize the cost of turns, the work [25] divides the region into cells and represents cells as a graph. The Dubins MCPP problem is formulated with the graph representation as a generalized traversal salesman problem (TSP). The exact coverage path is then obtained by applying the GTSP solver. Unfortunately, the work [25] is only applicable to a single robot. proposed an exact formulation to address the coverage problem for scattered regions. This formulation model MCPP as a $MinMax$ problem for balancing robot coverage times. However, this formulation does not apply to practical coverage applications since it does not consider robot kinetics. Several works models the optimal coverage problem as the $MinMax$ balanced and connected q-partition problem (BCP_q) and present various methods to resolve it.

Heuristic MCPP algorithms usually decompose the region into cells and represent them in a graph. With the graph representation, graph-partition and tree-partition strategies are utilized to divide the graph into multiple parts. Each part corresponds to one robot. The graph-partition strategy takes all information of the graph into account to obtain a (near-) optimal result. For example, to address the area patrolling problem of heterogeneous robots, the work [26] utilizes the auction algorithm to assign appropriate tasks to robots. Although the auction algorithm has the advantage of low complexity, its greedy strategy leads to local-optimal allocation. The authors in [18] extend the traditional market-based methods and propose a credit-based task allocation (CTA) algorithm. The CTA algorithm balances the tasks among robots by a credit model and reduces complexity by transforming a multi-objective optimization problem into a set of single-objective optimization problems. However, the CTA algorithm is unsuitable for coverage applications relying on Morse or BCD decomposition since it assumes that coverage tasks are uniform grids. In [2], two heuristics algorithms are presented to address the MCPP problem for known environments. The first algorithm calculates the Eulerian tour of visiting all tasks and produces k subtours by a k-postman approximation algorithm. The second one uses a greedy approach that divides the area into equal regions, covering each region with a single robot. Although the graph-partition strategy performs well, it has long runtime for graphs with a significant number of vertexes and edges.

The tree partition strategy reduces the runtime by deleting edges from the graph. However, some optimality is sacrificed since the search space shrinks after edge deletion. In [27], the authors proposed a spanning tree coverage (STC) algorithm for single robots. The STC algorithm incrementally builds a virtual tree and navigates the robot around the tree to achieve complete coverage. The work [28] extends the STC algorithm and presents a multi-robot STC algorithm, which reduces coverage time by two times while no repeated tasks are generated. Nevertheless, it cannot guarantee an optimal result with the rise of robots. Literature [29] proposed a polynomial-time algorithm that assigns tasks to k robots by finding a weighted tree of k (k = number of robots) covering all nodes. The polynomial-time algorithm ensures that its coverage time is eight times the optimal coverage time. However, it assumes that the trees can be overlapped. The genetic algorithm was also used to solve the tree partition problem due to its excellent performance. In the genetic algorithm, each individual is composed of a forest of non-intersecting trees, and the population evolves to find the (near-) optimum. For example, the work [20] presented a genetic algorithm based on tree partition and evolution. The genetic algorithm can handle graphs with up to 3000 nodes.[30] presents an algorithm called mofint for finding the least number of robots within a time limit. The mofint algorithm transforms the time-limit version of MCPP into a bi-objective optimization problem and applies a multi-objective genetic method. Although genetic algorithms perform well, they often produce local optimal solutions due to their evolutionary operations.

1.2. Dubins coverage

The exact and heuristic MCPP methods provide optimal or near-optimal paths that visit all points of the region. However, due to their lack of consideration of robot kinematics, several MCPP methods could not guarantee the path's curvature continuity. Curvature discontinuities threaten robot safety and degrade the robots' dead-reckoning abilities [5][31]. Thus, the construction of feasible and smooth paths has received much attention in robotic research fields [17].

The Dubins path [32] provides the shortest path for robots with a single forward speed and a maximum turning radius in open areas. As Dubins paths can be expressed analytically and are quickly computed, a series of Dubins coverage methods based on them were presented. Dubins coverage has numerous practical applications, such as automated agriculture [33], search and rescue, and seabed inspection. For example, the authors in [34] presented a coverage algorithm for a fixed-wing unmanned aerial vehicle (UAV). The coverage algorithm breaks the region into multiple subcells and produces the Eulerian circuit with minimum path repetition. The effectiveness of the proposed algorithm [34] has been validated in field trials. The authors in [25] modelled the Dubins coverage problem into an generalized traversal salesman problem (GTSP). The coverage path with the lowest non-working travel is obtained by transforming the GTSP into an asymmetry traversal salesman problem (ATSP). By re-setting the path cost between two points separated by obstacles, the work [35] extended [25] to non-convex environments. The work [4] proved that the optimal Dubins coverage problem is NP-complete and presents a coverage algorithm for single Dubins robot. In [36], the authors present two heuristics methods called DCRC and DCAC for addressing the CPP problem with multiple Dubins vehicles. DCRC generates an optimal Hamiltonian path and uses the route clustering to divide the path into K subpaths. DCAC divides the area into multiple partitions and applies the single-robot Dubins solver [4] to each subarea. The simulation results in [36] show that DCRC has better performance than DCAC.

2. System Overview

This section describes related definitions of the Dubins MCPP~~MDCPP~~ problem and presents a Dubins coverage framework.

2.1. Problem Statement

We assume to have K homogeneous Dubins robots to perform the coverage task. All robots constitute a robot set $R = \{r_1, ..., r_K\}$. The Dubins robots in R are equipped with the same task sensor for specific tasks (e.g., cleaning the floor or detecting objects). The task sensor can cover a rectangular area of w_1 in width. All robots start from the same starting point p_s and travel at a fixed speed s with a minimum turning radius r.

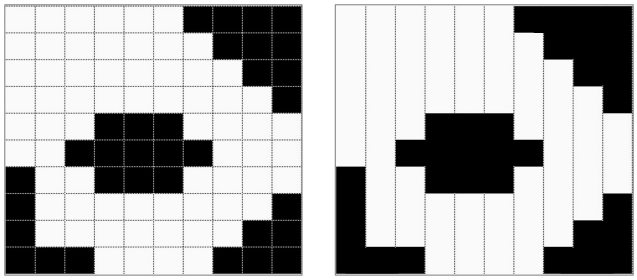


Figure 1. Decomposing the mission environment into cells.

The mission environment is assumed to be known and has been represented as a binary map. In this binary map, cells with values of 0 or 1 represent obstacles or allowed areas, respectively. To avoid being restricted to one kind of robot model, classical Dubins approaches [2][37] assume that obstacles are areas that are not necessarily covered but can be crossed. Similarly, this paper assumes that the robot can cross the obstacle. The semi-BCD decomposition [37] method is used to divide the region into N rectangular cells. All cells form the set of cells $C = \{c_1, ..., c_n, ..., c_N\}$, where c_n represents the n-th cell in C. Each cell has a width of w_1, and its height depends on the boundary of the region and obstacles. Fig. 1 represents an example of area decomposition.

The objective of MDCPP is to produce a path for each robot so that every point of the allowed areas is covered at least by one robot. Since all robots have the same kinetic constraints, an efficient solution is to minimize robot path lengths while equally distributing robot workloads. Hence, Dubins MCPP~~MDCPP~~ can be viewed as a *MinMax* problem, i.e., to minimize the maximum cost of all robots.

2.2. System Overview

This paper presents a Dubins coverage framework to address the Dubins MCPP ~~DMCPP~~ problem. As shown in Fig.2, the framework comprises coverage applications,

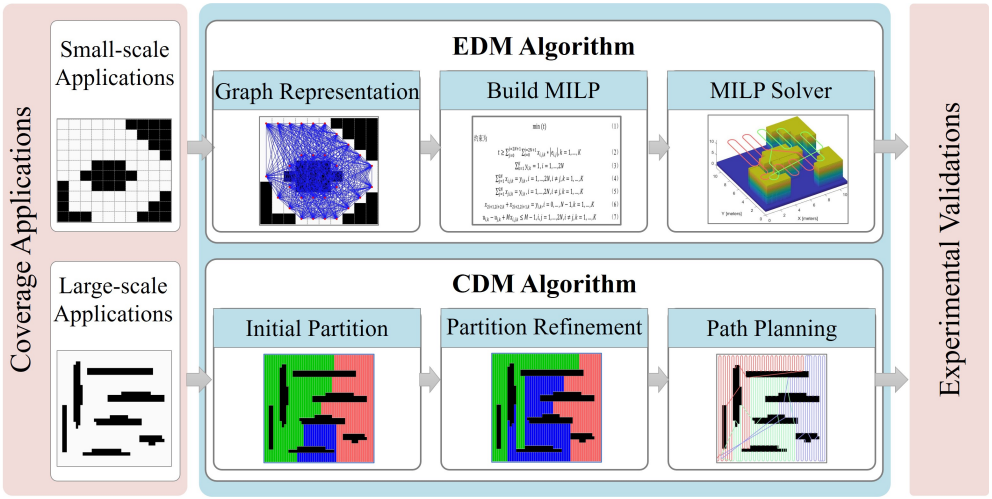


Figure 2. An overview of the Dubins coverage framework.

Dubins MCPP~~DCMPP~~ methods, and experimental validations. The component of the Dubins MCPP~~DMCPP~~ methods consists of an EDM~~exact EMD~~ algorithm and a heuristic CDM~~CMD~~ algorithm. The EDM algorithm~~EMD~~ represents coverage tasks as a graph and proposes an exact formation based on MILP. The MILP solver is used to produce the optimal Dubins coverage path by thoroughly searching the solution space.~~builds a MILP to obtain the optimal Dubins coverage paths.~~ The CDM algorithm~~CMD~~ divides the region into K subareas through initial partition and partition refinement modules. The single-robot Dubins solver [38] is then employed in each subarea. More details of EDM~~EMD~~ and CDM ~~CMD~~ are presented in Section 3 and 4.

3. Exact Dubin Multi-robot CPP (EDM)~~Multi-robot Dubin CPP (EMD)~~ Algorithm

Exact methods either provide an accurate partition or produce coverage paths without considering the turning cost of the robot covering a given region, resulting in a non-optimal coverage path. This paper presents an EDM algorithm to plan coverage paths. The EDM algorithm consists of two steps: graph representation and build MILP. The former step is to calculate the Dubins paths for covering coverage cells and turning from one cell to another. All Dubins paths will be represented as a connected graph. The latter step generates an exact formulation based on MILP to obtain the shortest Dubins coverage path. ~~produce precise K partitions of the region and employ a single-robot CPP for each partition. Exact partitions, however, may result in non-shortest coverage paths. This paper presents an EMD algorithm to produce shortest coverage paths in two steps. The first step is to represent coverage tasks in a connected graph. The second step is to build a MILP model to find the shortest path for each robot.~~

3.1. Graph Representation

Classical offline coverage methods decompose the region into cells and represent cells into a graph [39]. With the graph representation, the MCPP problem is transformed into TSP or Chinest Postman Problem to obtain the fastest or shortest path [25][2]. As most offline MCPP methods do, the EDM algorithm divides the mission environment ~~As mentioned before, the mission environment has been divided~~ into a set of cells (i.e., C). Each cell in C consists of two endpoints and a line segment connecting them. As shown in 3(a), the robot can either enter the cell from the top endpoint and cover it from the top down or enter the cell from the bottom endpoint and cover it from the bottom up. N cells of C correspond to $2N$ endpoints, which constitute the set of endpoints $P = \{p_1, ..., p_{2N}\}$. Each pair of endpoints p_{2n+1} and p_{2n+2} indicate the upper and lower endpoint of $c_n, 0 \leq n \leq N-1$, respectively. All endpoints in P are represented as a connected graph $G = (V, E)$, where V and E refer to vertex and edge sets, respectively. V consists of $2N+1$ vertexes, where the first vertex v_0 corresponds to the starting point p_s, and other vertexes $v_n, n = 1, .., 2N$ represent the n-th endpoint p_n in P. Each edge $e_{i,j} \in E$ indicates the Dubins path between the vertex v_i and v_j. Different from the graph presented in [25][2][39], E consists of the Dubins paths for the robot turning and covering the cell.

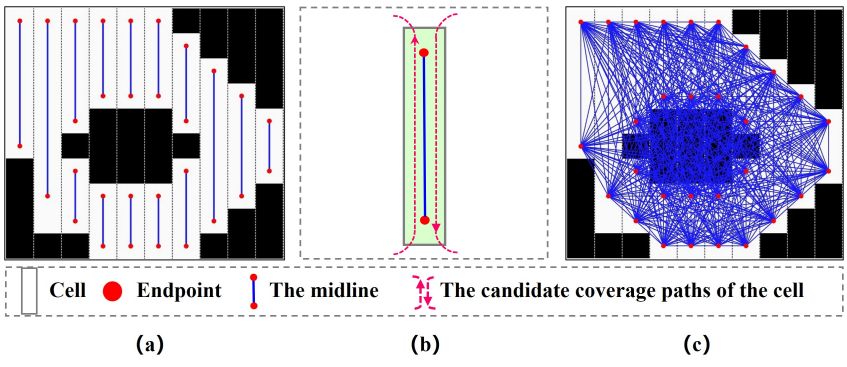
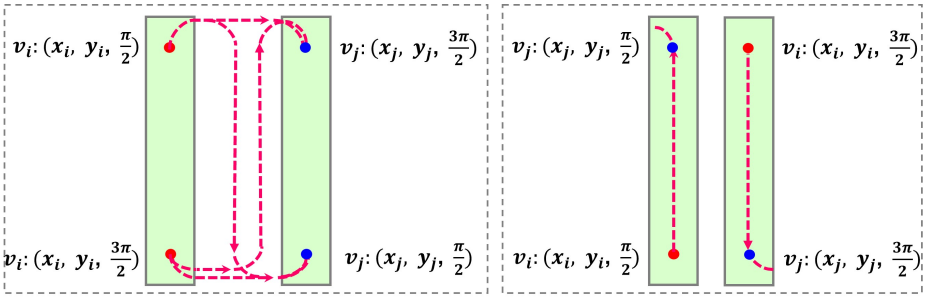


Figure 3. Graph representation. (a) Endpoints of each cell. (b) Coverage paths of each cell. (c) Graph.

The edge between v_i and v_j represents the Dubins path from the start pose $v_i : (x_i, y_i, \theta_i)$ to the target pose $v_j : (x_j, y_j, \theta_j)$. The x/y coordinates (x_i, y_i) and (x_j, y_j) are fixed, but the angle θ_i and θ_j depends on v_i and v_j's relative positions. There are two cases for θ_i and θ_j. In the first case, v_i and v_j belong to the same cell. Suppose that v_i is the lower endpoint of the cell. The robot enters the cell from v_i and covers the entire cell from bottom to top. Thus, θ_i and θ_j are set as $\frac{\pi}{2}$; alternatively, $\theta_i = \theta_j = \frac{3\pi}{2}$. The second case is that v_i and v_j belong to different cells. θ_i will be set to $\frac{3\pi}{2}$ if v_i is the lower endpoint of the cell (i.e., the robot leaves the cell from its lower endpoint); otherwise, $\theta_i = \frac{\pi}{2}$. Similarly, θ_j will set $\frac{\pi}{2}$ if v_j is the lower endpoint of the cell (i.e., the robot enters the cell from its lower endpoint); otherwise, $\theta_j = \frac{3\pi}{2}$. Fig.4 shows an example of how to calculate the angle. After determining the start and target pose, the Dubins path between v_i and v_j is calculated. The length of the Dubins path is set as the weight of $e_{i,j}$.



(a) v_i **and** v_j **belong to different cells** **(b)** v_i **and** v_j **belong to the same cell**

Figure 4. The start pose $v_i : (x_i, y_i, \theta_i)$ and target pose $v_j : (x_j, y_j, \theta_j)$.

3.2. Build MILP

The optimal coverage aims to find K tours that start and end at the depot p_s, so every point of allowed areas is visited, and the maximum cost of K tours is minimized. Thus, EDME~~M~~D models the Dubins MCPP problem as a MILP with the *MinMax* objective of minimizing the longest path cost among K robot paths. Define t as the cost of the longest Dubins path. The objective is

$$min(t) \tag{1}$$

s.t.,

$$t \geq \sum_{j=0}^{2N+1} \sum_{i=0}^{2N+1} x_{i,j,k} |e_{i,j}|, k = 1, ..., K \tag{2}$$

$$\sum_{k=1}^{K} y_{i,k} = 1, i = 1, .., 2N \tag{3}$$

$$\sum_{j=1}^{2N} x_{i,j,k} = y_{i,k}, i = 1, .., 2N, i \neq j, k = 1, ..., K \tag{4}$$

$$\sum_{j=1}^{2N} x_{j,i,k} = y_{i,k}, i = 1, .., 2N, i \neq j, k = 1, ..., K \tag{5}$$

$$\begin{aligned} x_{2i+1,2i+2,k} + x_{2i+2,2i+1,k} &= y_{i,k} \\ i = 0, .., N-1, k &= 1, ..., K \end{aligned} \tag{6}$$

$$\begin{aligned} u_{i,k} - u_{j,k} + M x_{i,j,k} &\leq M - 1 \\ i, j = 1, .., 2N, i \neq j, k &= 1, ..., K \end{aligned} \tag{7}$$

$$x_{i,j,k} \in \{0,1\}, y_{i,k} \in \{0,1\}, \forall i,j,k \tag{8}$$

where $x_{i,j,k} = 1$ represents that the robot r_k visits vertex j immediately after vertex i; otherwise, $x_{i,j,k} = 0$. $y_{i,k} = 1$ indicates that the robot r_k visits vertex i; otherwise, $y_{i,k} = 0$. Eq.(3) states that each vertex should be visited exactly once. Eq.(4)-(5) ensures that once a robot visits a vertex, it must also depart from the same vertex. Eq.(6) ensures that two endpoints of a cell should be traversed by sequence. Eq.(7) is the MTZ-based subtour elimination constraints [40]. The MILP is an extension of asymmetry multiple traveling salesman problems (MTSP).

3.3. Pseudo-code of the EDM~~EMD~~ algorithm

Algorithm 1 shows the pseudo-code of the EDM~~EMD~~ algorithm. It inputs the region ($\mathcal{A}$), the robot set (R) and starting point (p_s) and outputs K coverage paths $\{P_1, ..., P_K\}$. First, K coverage paths are initialized as empty (Line 1), and the region $\mathcal{A}$ has been divided into a set of cells C (Line 2). The cell set C is represented as a graph G (Line 3), and the cost matrix associated with G is calculated (Line 4). EDM~~EMD~~ models the Dubins MCPP problem as a MILP (Line 5) and utilizes the MILP Solver [41] to obtain the traversal sequence $\{T_1, ..., T_K\}$ (Line 6). Coverage paths are calculated according to $\{T_1, ..., T_K\}$ (Line 7).

Algorithm 1 EDM Algorithm

Input: $\mathcal{A}, \mathcal{R}, p_s$
Output: $P_1, ..., P_K$

1: Initialize: $P_1, ..., P_K \leftarrow \emptyset$;
2: $C \leftarrow$ Area_Decomposition($\mathcal{A}$);
3: $G \leftarrow$ Graph_Representation(C, p_s);
4: $CostMatrix \leftarrow$ calculate the cost between points in G
5: Build_MILP(G); // Eq.(1) - Eq.(7)
6: $\{T_1, ..., T_K\} \leftarrow$ MILP_Solver;
7: $\{P_1, ..., P_K\} \leftarrow$ Dubins_Solver($\{V_1, ..., V_K\}$);
8: **return** $P_1, ..., P_K$

Computational Complexity: Let M be the number of vertexes in G. The cost matrix can be calculated in $\mathcal{O}(M^2)$ times. The asymmetry MTSP can be transformed into an asymmetry TSP, which tasks $\mathcal{O}(M^3)$ times in the worst case [42]. Thus, the overall complexity of EDM ~~EMD~~ is $\mathcal{O}(M^3)$.

4. Heuristic Credit-based Dubin Multi-robot CPP(CDM)~~Multi-robot Dubin CPP (CMD)~~ Algorithm

EDM~~EMD~~ algorithm provides an effective weapon to plan exact coverage paths for small-scale coverage applications. However, several coverage applications involve a large number of coverage tasks and allow for near-optimal results. Therefore, this paper presents a heuristic CDM~~CMD~~ algorithm consisting of three components: initial partition, partition refinement, and path planning. The initial partition component utilizes the regional growth strategy to divide the region into K subareas. The partition refinement component balances K subareas by a tree-partition strategy, and the path planning component employs the single-robot Dubins solver [38] to each subarea.

4.1. Initial Partition

As mentioned in 2.1, the mission environment has been divided into a set of cells C. The initial partition component represents C as a connected graph $G1 = \{V1, E1\}$, where the vertex represents the cell, and the edge represents the common border between cells. Fig.5 shows an example of the graph representation. The region growth strategy based on the credit model [18] is used to divide the region (i.e., the graph $G1$) into K partitions.

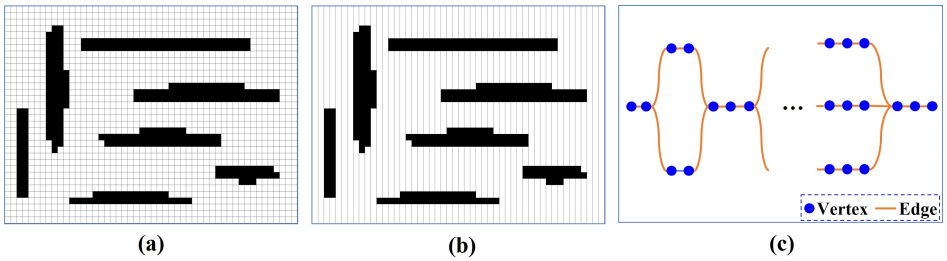


Figure 5. Graph representation. (a) The input map. (b) Coverage cells. (c) The connected graph.

In the credit model, K partitions act as traders in the virtual economy. Coverage cells are tradable commodities with measurable values. Additionally, a virtual bank is introduced. Traders can open credit accounts with a balance equal to $\frac{w(V1)}{K}$. The virtual bank maintains accounts and manages assets for traders. Each trader can borrow from the bank (without interest) if his account balance reaches zero. When a trader buys a cell v_t, its account balance is reduced by $|v_t|$, where $|v_t|$ represents the area of the cell v_t. On the other hand, if a trader sells a cell v_t, the account balance increases by $|v_t|$. Traders continue to buy cells, and the corresponding account balance decreases.

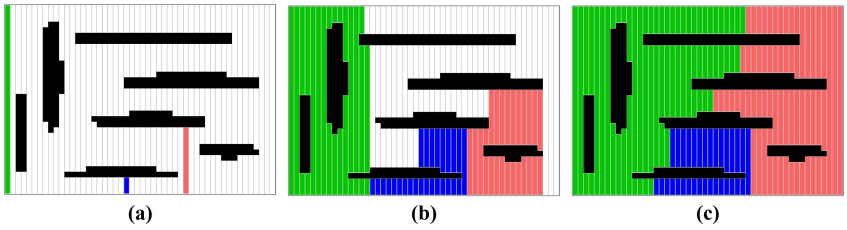


Figure 6. An example of the initial partition for three robots.

The initial partition component uses the regional growth strategy to divide the region into K partitions. First, cells in C are sorted in increasing order by the x-coordinate, followed by the y-coordinate, resulting in a sequence of cells S. The K cells, distributed at equal intervals in S, are set as the seeds of the K partitions. Second, every partition alternately buys cells and grows around the seed as the number of bought cells increases. All partitions become larger and larger until all cells in $G1$ have been purchased. In this case, $G1$ has been divided into K partitions. Fig.6 shows an example of the region growth strategy.

4.2. Partition Refinement

Due to complex obstacles, the initial K partitions may not be balanced. In order to obtain a balanced result, the partition refinement component reallocates tasks among partitions by way of task transactions. Each task transaction is performed in three steps. The first step is to determine the two partitions for the task transaction. The k-th partition (i.e., V_k) with the largest account balance is set as the buyer, i.e., the partition that receives tasks. Let ADV be the set of partitions adjacent to V_k. The partition $u \in ADV$ that has the greatest difference with the account balance of V_k becomes the seller, i.e., the partition that dispatchs tasks. $found(k)$ and $found(u)$ represents the account balance of the seller and buyer, respectively.

The second step is to decide which tasks the seller and buyer will trade. Suppose that $AC \in V_u$ is the cell set that shares a common border with V_k. The cell v_m with the biggest weight in AC is selected for the candidate trade task EV. There are two possible cases, depending on the connection between v_m and V_u. In the first case, v_m is not the cut point of V_u. The trade task EV is set as v_m since both seller and buyer remain connected after the task transaction. The second case is that v_m is the cut point of V_u (i.e., removing v_m disconnects V_u). The seller V_u becomes disconnected if it sells v_m to V_k. However, disconnect partitions cause robot collisions and complicate robot control [30]. Thus, the depth first search (DFS) method is utilized to find the Q subtrees $\{V_{u,1}, ...V_{u,Q}\}$ in V_u whose

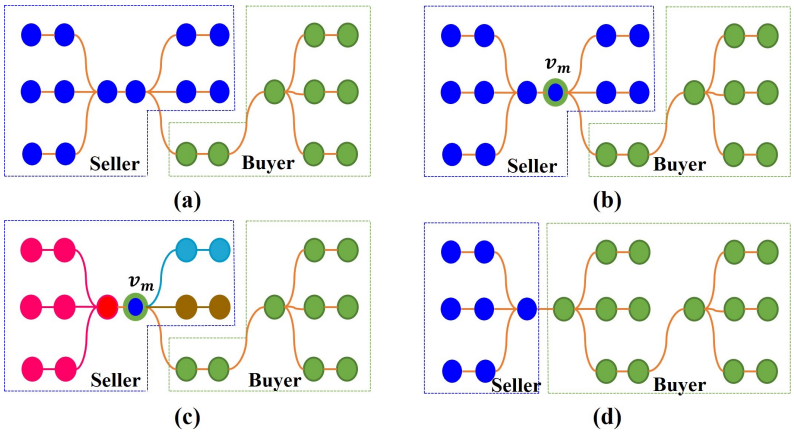


Figure 7. An example of the tree-partition strategy. (a) The graphs of the seller and buyer. (b) The adjacent vertex v_m. (c) Three subtrees with the root v_m in the seller. (d) The buyer and seller after the task transcation.

root nodes are v_m. $V_{u,i}$ and $V_{u,j}, i \neq j, i, j = 1, ..., Q$ will be disconnected if v_m is removed from V_u. In order to maintain its connectivity, the seller V_u needs to reserve one subtree and set the other tasks as trade tasks. In order to determine which subtree the seller retains, a transaction index is defined, which quantifies the balance between the buyer and seller's tasks. Suppose the seller reserves the q-th subtree $V_{u,q}$. The seller and the buyer will be updated to $V_{k'} = V_k \cup (V_u - V_{u,q})$ and $V_{u'} = V_{u,q}$, respectively. The trade index δ_q of the q-th subtree is set as $max(abs(found(k'), found(u')))$, where $found(k')$ and $found(u')$ represents the account balance of $V_{k'}$ and $V_{u'}$. Q subtrees correspond to Q trade indexes $\{\delta_1, ..., \delta_Q\}$. The smaller the trade index, the more balanced the buyer and seller. Let δ_{q1} be the minimum of $\{\delta_1, ..., \delta_Q\}$, and δ_B be the trade index of V_k and V_u. If $\delta_{q1} < \delta_B$, the seller retains the $V_{u,q1}$ with the least transaction index. The remaining tasks $V_u - V_{u,q1}$ are set as the trade tasks, i.e., $EV = V_u - V_{u,q1}$. Fig.7~~6~~shows an example of the tree-partition strategy. Alternatively, $\delta_{q1} > \delta_B$ indicates that the tasks of the seller and the buyer do not become balanced after the task transaction. A new v_m from AC is set as the candidate trade task EV, and the tree-partition strategy is applied for the new v_m. If all cells in AC can not provide more balance partitions, a new task transaction is performed since V_u and V_k are a pair of non-tradable partitions. With the tree-partition strategy, a set of tasks rather than a single task are reallocated, while keeping the connectivity of partitions.

In the third step, the buyer and seller trade tasks and update their account balances. The buyer V_k purchases the task set EV, and its account balance becomes $w(V_k) - w(EV) + D_{s,k}$, where $w(EV)$ and $D_{s,k}$ represent the sum of weights of EV and the shortest distance between the starting point p_s and V_k. $D_{s,k}$ is calculated so the farther-distance traveling robot is compensated by assigning fewer tasks instead of dividing the region into K equal sections. Similarly, the seller V_u sells the task set EV and adjusts its account balance to $w(V_u) + w(EV) + D_{s,u}$, where $D_{s,u}$ represents the shortest distance between the starting point p_s and V_u.

With the completion of task transactions, the K partitions become more and more balanced. As soon as the number of task transactions reaches the preset upper limit, the partition refinement component ends and returns K partitions $\{V_1, ..., V_K\}$.

4.3. Path Planning

After receiving K partitions from the partition refinement component, the path planning component applies the single-robot Dubins solver [37] to each partition. A set of K Dubins coverage paths is generated, and each one corresponding to one robot. The complete coverage is achieved if each robot moves along the corresponding coverage path. Fig.8~~7~~ shows an example of the CDM~~CMD~~ algorithm.

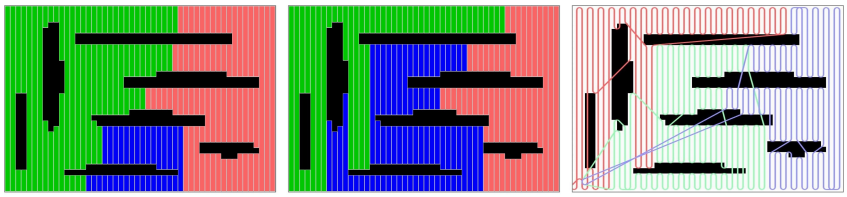


Figure 8. (a) Initial partition. (b) Partition refinement. (c) Path Planning.

4.4. Pseudo-code of the CDM~~CMD~~ algorithm

Algorithm 2 shows the pseudo-code of the CDM~~CMD~~ algorithm. It decomposes the region $\mathcal{A}$ into a set of cells C and represents all cells into a graph $G1$ (Lines 2-3). The graph G is divided into K partitions by the initial partition (Line 4). These K partitions are refined by task transactions (Lines 5-23). For each task transaction, the buyer V_k and the seller V_u are determined, followed by the set of adjacent cells AC between V_k and V_u (Lines 7-8). The seller V_u is the partition that is adjacent to and can trade with V_k. For each cell in AC, the tree-partition strategy calculates the corresponding trade tasks EV (Line 11). If $EV \neq \emptyset$, V_k and V_u trade tasks and updates their account balances (Lines 13-14). The symbol $succeed$, which indicates the success of the task transaction, is marked as $true$ (Line 15). If $succeed$ remains $false$, V_k and V_u are marked as a pair of non-tradable partitions (Lines 19-21). Upon the number of task transactions equaling $MaxI$, K partitions $\{P_1, ..., P_K\}$ is obtained. Next, the single-robot Dubins solver [38] is used for each partition to generate coverage paths $\{P_1, ..., P_K\}$ (Lines 24).

Algorithm 2 CDM Algorithm

Input: $\mathcal{A}, \mathcal{R}, p_s$
Parameter: $MaxI$: The maximum number of task transactions
Output: $P_1, ..., P_K$

1: Initialize: $P_1, ..., P_K \leftarrow \emptyset$;
2: $C \leftarrow$ Area_Decomposition($\mathcal{A}$);
3: $G1 \leftarrow$ Graph_Representation(C, p_s);
4: $found, V_1, ..., V_K \leftarrow$ initial_partition($G1, s$);
5: $count \leftarrow 0$;
6: **while** $count < MaxI$ **do**
7: $V_k, V_u \leftarrow$ determine the seller and the buyer;
8: $AC \leftarrow$ calculate the set of adjacent cells between V_k and V_u;
9: $succeed \leftarrow false$;
10: **for** each v_m in AC **do**
11: $EV \leftarrow$ tree_partition(V_k, V_u, v_m);
12: **if** $EV \neq \emptyset$ **then**
13: Trade tasks EV;
14: Update $found$;
15: $succeed \leftarrow true$;
16: break;
17: **end if**
18: **end for**
19: **if** $succeed = false$ **then**
20: Mark V_k and V_u as a pair of non-tradable partitions.
21: **end if**
22: $count++$;
23: **end while**
24: $\{P_1, ..., P_K\} \leftarrow$ Dubins_Solver($\{V_1, ..., V_K\}$);
25: **return** $P_1, ..., P_K$;

Computational Complexity: Let M be the number of cells in C. The initial partition component takes $\mathcal{O}(M)$ times. The complexity of the partition refinement component is

$\mathcal{O}(M * MaxI)$ in the worst case, but the worst cases are scarce. The Dubins solver takes $\mathcal{O}(M^3)$ times to calculate the Dubins path[42]. Thus, the overall complexity of the CDM ~~CMD~~ algorithm is $\mathcal{O}(M^3)$.

5. Experiments

The computational experiments were carried out on a PC with Intel(R) Core(TM) CPU i5-8300H, 2.30 GHz processor, 16G RAM, WIN 10. All experiments were performed on Dubins robots with kinematic constraints such as a forward speed of 1.0 m/s and a minimum turning radius of 1 m. A task sensor with a detection range of 1 m was incorporated into each robot. First, to demonstrate the superiority of the proposed algorithms, comparison experiments with exact and heuristic algorithms were conducted on different size maps. Second, simulation experiments based on a high-fidelity UAV model [43] were conducted to verify the feasibility of EDM~~EMD~~ and CDM~~CMD~~.

5.1. Comparison experiments in small scenes

The first level of validation was done via simulations on four small scenes with 10 m $\times$ 10 m $\times$ 10 m. Fig.9 demonstrates the point cloud maps of four scenes, which contain several obstacles with irregular shapes and same heights. For each scene, Dubins robots start and end at the same starting point, located in the bottom left corner of the map. EDM~~EMD~~ and CDM~~CMD~~ were compared with the exact Milpflow algorithm [20] and heuristic DCRC algorithm [36]. Milpflow provides a precise area-division result instead of coverage paths. In order to achieve a fair comparison, the state-of-art Dubins solver [38] is employed to plan Dubins path for Milpflow. DCRC generates an optimal Hamiltonian path and divides the path into K subpaths. Exact Mofint and EDM~~EMD~~ algorithms utilize the Gurobi optimization tool [41] to obtain the optimal solution, and their optimization time is uniformly set as 1200 s.

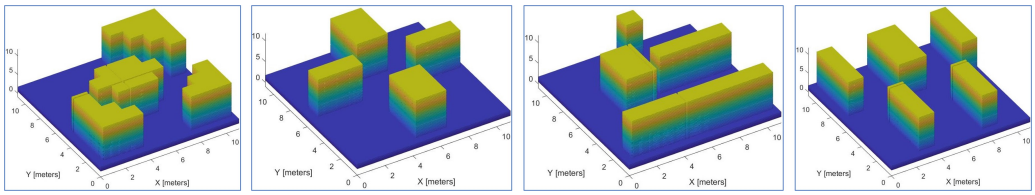


Figure 9. The four point-cloud maps where EDM~~EMD~~ and CDM~~CMD~~ were tested. Each environment is with the size of 10 m $\times$ 10 m $\times$ 10 m.

A variety of experiments were performed using teams of two or three robots on different maps. Fig.10 and 11 demonstrate snapshots of coverage paths produced by Milpflow [20], DCRC [36], EDM~~EMD~~, and CDM~~CMD~~, respectively. These snapshots show that Milpflow and CDM produce relatively concentrated paths for every robot since they allocate a set of connected coverage cells to every robot. In contrast to Milpflow and CDM, EDM and DCRC generate the single-robot coverage path that is not limited in a particular area.~~a single robot has a relatively concentrated coverage path in the Milpflow and CMD algorithms since they divide the region into subareas and then apply a single-robot Dubins solver to each subarea. The DCRC algorithm generates K coverage paths by splitting the single-robot tour into K subtours, while EMD optimizes the MILP to produce optimal coverage paths. Thus, their single-robot coverage paths are not clustered in specific areas.~~

Fig.12 compares coverage times of Milpflow [20], DCRC [36], EDM~~EMD~~, and CDM ~~CMD~~, respectively. The comparison results show that, compared with heuristic DCRC and CDM, Milpflow and EDM provide fewer coverage times by thoroughly searching the solution space. Furthermore, EDM produces the least coverage times in all scenes because it generates the optimal Dubins coverage path rather than the area division provided by Milpflow.~~The comparison results show that EMD provides the least coverage time than other exact or heuristic coverage algorithms. However, EMD takes more runtime than the other three algorithms since it has a large search space.~~

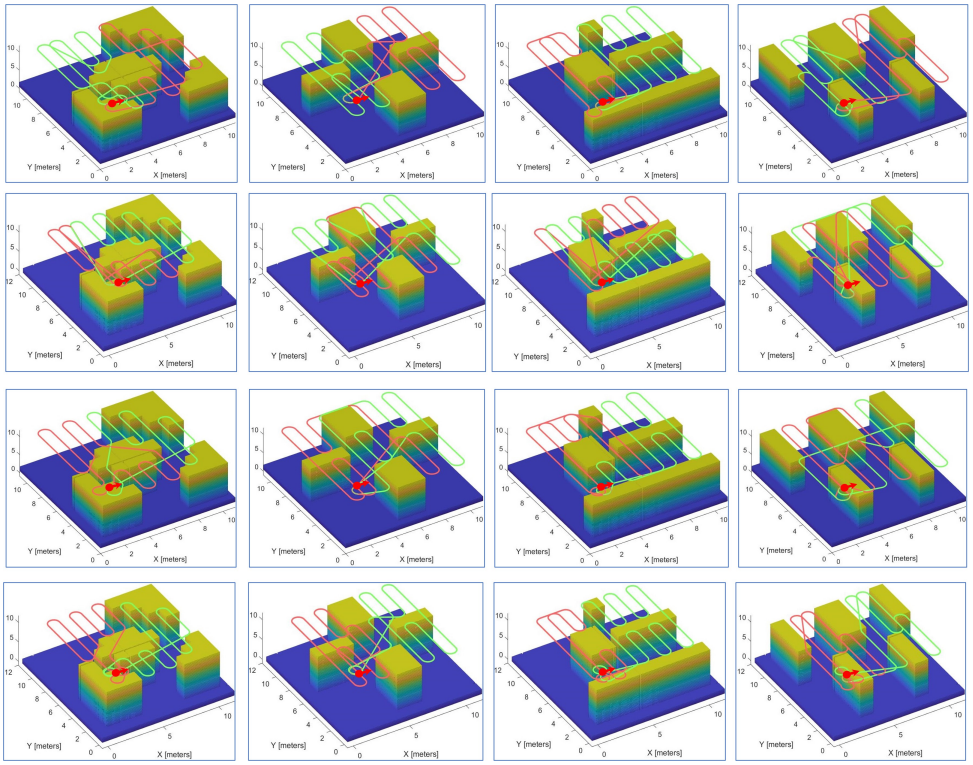


Figure 10. Simulation instances with two robots. The first to fourth rows represent the snapshots of coverage paths provided by Milpflow [20], DCRC [36], EDM~~EMD~~, and CDM~~CMD~~, respectively.

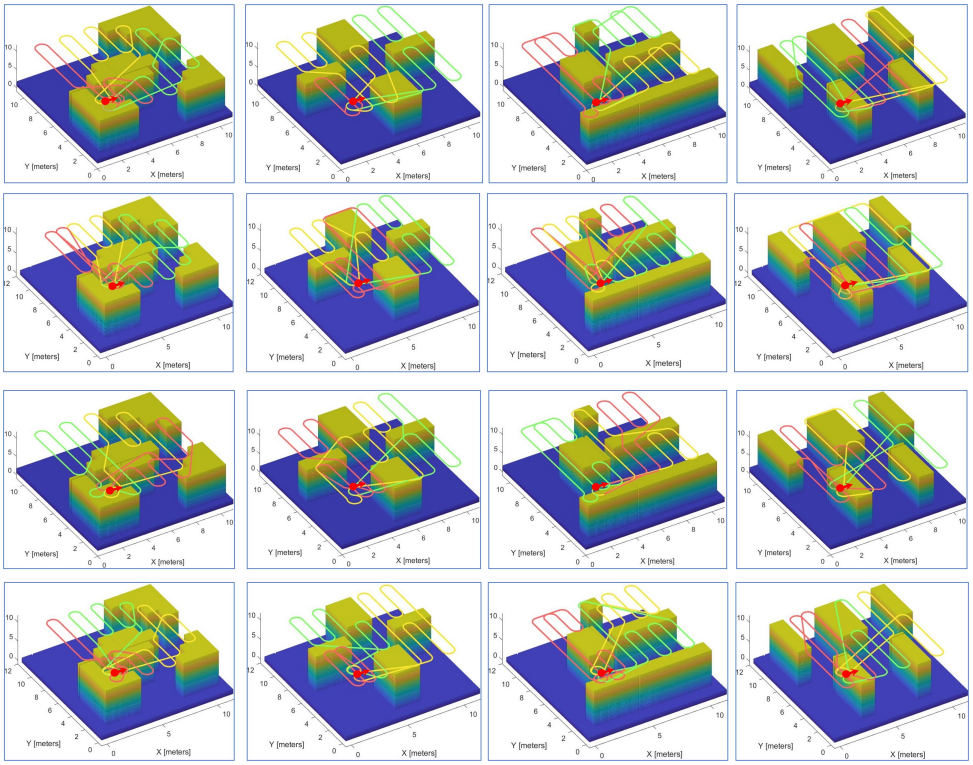
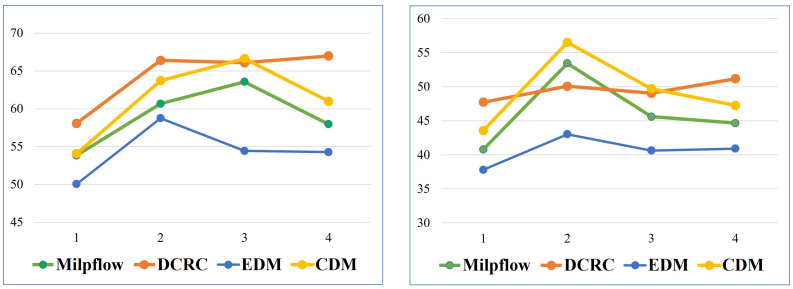


Figure 11. Simulation instances with three robots. The first to fourth rows represent the snapshots of coverage paths provided by Milpflow [20], DCRC [36], EDM~~EMD~~, and CDM~~CMD~~, respectively.

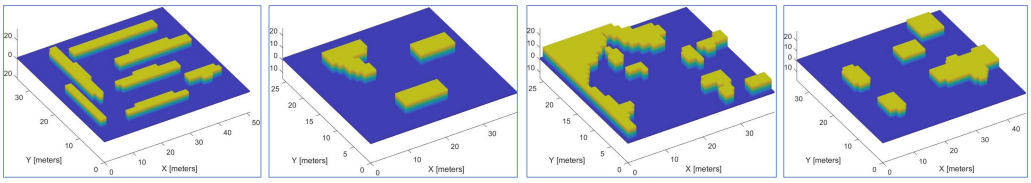


（a） Robot number = 2 （b） Robot number = 3

Figure 12. The comparison of coverage times of Milpflow [20], DCRC [36], EDM~~EMD~~, and CDM ~~CMD~~. Fewer coverage times are better.

5.2. Comparison experiments in large scenes

In order to evaluate the performance of the proposed algorithm, a variant of the well-known environments from [2] was used. As shown in Fig.13, the maps differ in terms of sizes and shapes. A set of experiments were conducted with teams of $\{3, 6, 9, 12\}$ robots. Since Milpflow and EDM~~EMD~~ cannot provide efficient solutions within limited time, this subsection only evaluates the heuristic CDM~~CMD~~ and DCRC [36] algorithms. Two metrics are used for performance evaluation as follows:(i) *coverage time*, and (ii) *computation time*.



（a） （b） （c） （d）

Figure 13. Four point cloud maps where CDM~~CMD~~ and DCRC were tested. (a) Multi-cell(34 m $\times$ 50 m $\times$ 10 m); (b) Farm(25 $\times$ 38 m $\times$ 10 m); (c) Rural Quebec(25 m $\times$ 38 m $\times$ 10 m); (d) Cave(34 m $\times$ 45 m $\times$ 10 m);

Fig.16 and 17 demonstrate snapshots of coverage paths generated by DCRC and CDM for 3 and 6 robots, respectively. These snapshots show that the CDM algorithm provides a set of connected cells for every robot, while a single robot's coverage cells in DCRC may be disconnected. Paths between disconnected cells probably revisit the covered area, which increases the coverage time. Indeed, as illustrated in Fig.14, the CDM algorithm provides fewer coverage times than DCRC in most experiments.

Fig.15 shows the computation times of CDM~~CMD~~ and DCRC with $\{3, 6, 9, 12\}$ robots, respectively. It is observed that DCRC provides a approximately equal computation time in each scene, while the computation time of CDM~~CMD~~ decreases with the increase of robot number. The difference in computation time between CDM~~CMD~~ and DCRC derives from the search space. The larger the search space, the longer the computation time of the algorithm. DCRC plans a single-robot coverage path in terms of the entire map, which corresponds to a large search space. In contrast, CDM~~CMD~~ divides the map into K subareas and plans the path for each subarea. Compared with the entire map, subareas corresponds to a small search space. With the increase in robot number, CDM~~CMD~~'s computation times become smaller.

~~Fig.16 and 17 demonstrate snapshots of coverage paths generated by DCRC and CDMCMD for 3 and 6 robots, respectively. DCRC divides the single-robot optimal route into K subpaths, each corresponding to a robot. As shown in the first row of Fig.16 and 17, crossings between subpaths occur in DCRC. In contrast, CDMCMD divides the map into K subareas and employs the single-robot Dubins solver [38] for each subarea. Thus, each robot's coverage path is clustered within its subarea.~~

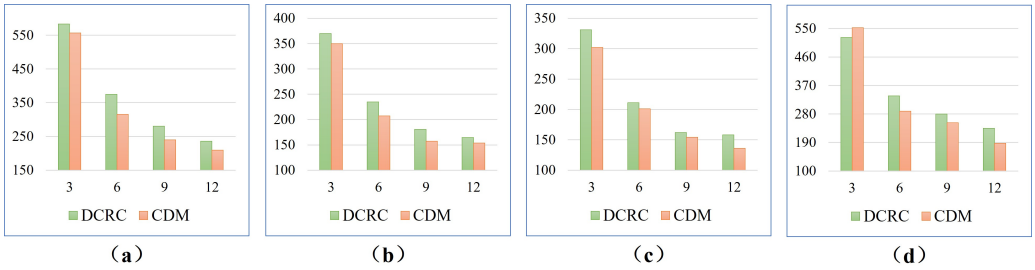


Figure 14. The comparison of coverage times for four different environments. Less coverage times are better.

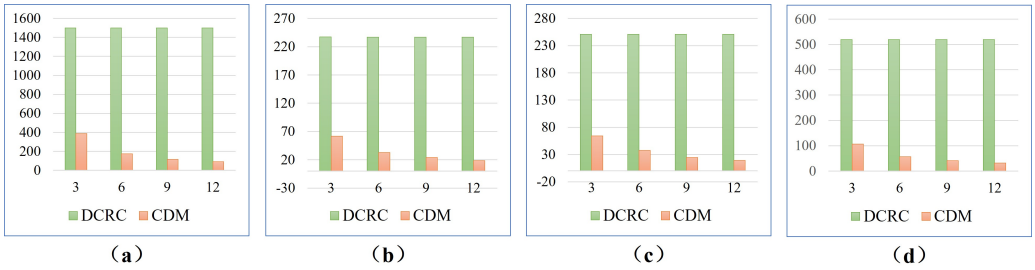


Figure 15. The comparison of computation times for four different environments. Less computation times are better.

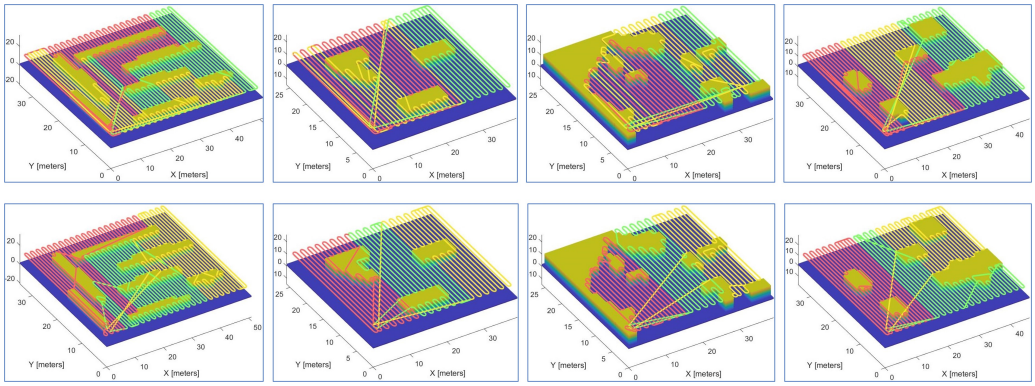


Figure 16. Coverage paths of DCRC (first row) and CDM~~CMD~~ (second row) with 3 robots.

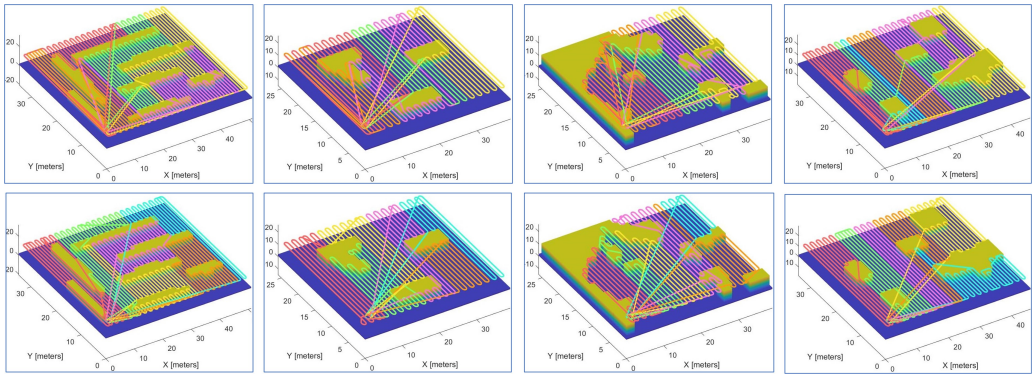


Figure 17. Coverage paths of DCRC (first row) and CDM~~CMD~~ (second row) with 6 robots.

5.3. Feasibility experiments of EDM~~EMD~~ and CDM~~CMD~~

We validate EDM~~EMD~~ and CDM~~CMD~~ algorithms with a high-fidelity fixed-wing UAV model [43] in Simulink. A waypoint follower is integrated into the fixed-wing UAV model, which calculates the desired heading based on the current pose, look-ahead distance, and coverage paths. Experiments were conducted on UAVs with kinematic constraints such as 0.5m turning radius and 1 m/s speed. Each UAV was set at a different flight height to ensure its safety. Fig.18 and 19 demonstrate snapshots of the simulated UAV paths for EDM~~EMD~~ and CDM~~CMD~~ , respectively. The snapshots show that EDM~~EMD~~ and CDM ~~CMD~~ are applicable to fixed-wing UAVs.

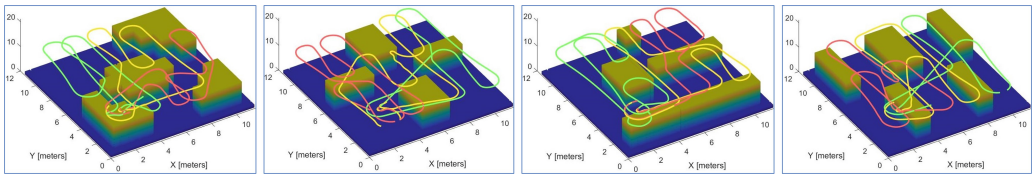


Figure 18. UAV simulated paths of EDM~~EMD~~ with 3 robots.

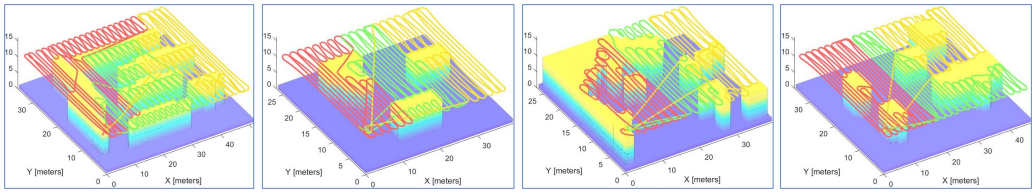


Figure 19. UAV simulated paths of CDM~~CMD~~ with 3 robots.

6. Conclusion

This paper presents an EDM~~exact EMD~~ algorithm and a heuristic CDM~~CMD~~ algorithm to address the Dubins MCPP~~DMCPP~~ problem. EDM~~EMD~~ formulates the Dubins MCPP~~DMCPP~~ problem into a MILP to produce the shortest Dubins coverage paths. CDM ~~CMD~~ balances the coverage tasks among robots by a credit model and reduces the complexity of the Dubins MCPP~~DMCPP~~ problem by a tree partition strategy, providing an approximate optimal solution. It is shown that both EDM and CDM can provide smooth and continuous Dubins coverage paths. Comparison experiments with other exact or heuristic algorithms demonstrate that EDM produces the fastest Dubins coverage path in small-scale scenes, and CDM produces less coverage time and shorter computation time than other heuristic algorithms in large-scale scenes. Feasibility experiments show that the results from the simulations and the analysis performed on those results hold for high-fidelity Dubins robotic systems.~~EMD and CMD were validated in comparison experiments with other exact and approximate algorithms. The comparison results show that EMD provides the least coverage time in small-scale scenes, while CMD produces less coverage time and shorter computation time in large-scale scenes. In addition, the applicability of EMD and CMD was validated via high-fidelity simulations for fixed-wing UAVs.~~ Future research areas include: (i) extending online coverage to unknown environments, (ii) applying to real Dubins robots.

Author Contributions: The work presented here was carried out in collaboration among all authors. All authors have contributed to, seen and approved the manuscript.

Acknowledgments: This work was supported by Science and Technology Innovation 2030 Major Project under Grant No.2020AAA0104802. The work was also supported by National Natural Science Foundation of China (Grant No. 91948303). The authors would like to thank the anonymous reviewers for their valuable suggestions and providing many possible directions for the future work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fevgas, G.; Lagkas, T.; Argyriou, V.; Sarigiannidis, P. Coverage path planning methods focusing on energy efficient and cooperative strategies for unmanned aerial vehicles. *Sensors* **2022**, *22*, 1235.
2. Karapetyan, N.; Moulton, J.; Lewis, J.S.; Li, A.Q.; O'Kane, J.M.; Rekleitis, I. Multi-robot dubins coverage with autonomous surface vehicles. In Proceedings of the International Conference on Robotics and Automation (ICRA), Brisbane, Australia. IEEE, 2018, pp. 2373–2379.
3. Chen, J.; Du, C.; Zhang, Y.; Han, P.; Wei, W. A clustering-based coverage path planning method for autonomous heterogeneous UAVs. *IEEE Transactions on Intelligent Transportation Systems* **2021**.
4. Lewis, J.S.; Edwards, W.; Benson, K.; Rekleitis, I.; O'Kane, J.M. Semi-boustrophedon coverage with a dubins vehicle. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 5630–5637.
5. Wang, X.; Jiang, P.; Li, D.; Sun, T. Curvature continuous and bounded path planning for fixed-wing UAVs. *Sensors* **2017**, *17*, 2155.
6. Kan, X.; Teng, H.; Karydis, K. Multi-robot field exploration in hex-decomposed environments for dubins vehicles. In Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dali, China. IEEE, 2019, pp. 449–455.
7. Coombes, M.; Chen, W.H.; Liu, C. Flight testing Boustrophedon coverage path planning for fixed wing UAVs in wind. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 711–717.
8. Wilson, J.P.; Mittal, K.; Gupta, S. Novel motion models for time-optimal risk-aware motion planning for variable-speed AUVs. In Proceedings of the OCEANS 2019 MTS/IEEE SEATTLE. IEEE, 2019, pp. 1–5.
9. Maini, P.; Gonultas, B.M.; Isler, V. Online coverage planning for an autonomous weed mowing robot with curvature constraints. *IEEE Robotics and Automation Letters* **2022**, *7*, 5445–5452.
10. Deng, D.; Jing, W.; Fu, Y.; Huang, Z.; Liu, J.; Shimada, K. Constrained heterogeneous vehicle path planning for large-area coverage. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China. IEEE, 2019, pp. 4113–4120.
11. Muñoz, J.; López, B.; Quevedo, F.; Monje, C.A.; Garrido, S.; Moreno, L.E. Multi UAV Coverage Path Planning in Urban Environments. *Sensors* **2021**, *21*, 7365.
12. Rekleitis, I.; New, A.P.; Rankin, E.S.; Choset, H. Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence* **2008**, *52*, 109–142.
13. 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* **2021**.
14. Cabreira, T.M.; Brisolara, L.B.; Ferreira Jr, P.R. Survey on coverage path planning with unmanned aerial vehicles. *Drones* **2019**, *3*, 4.
15. Aggarwal, S.; Kumar, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications* **2020**, *149*, 270–299.
16. Yu, X.; Jin, S.; Shi, D.; Li, L.; Kang, Y.; Zou, J. Balanced multi-region coverage path planning for unmanned aerial vehicles. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2020, pp. 3499–3506.
17. Khan, A.; Noreen, I.; Habib, Z. On Complete Coverage Path Planning Algorithms for Non-holonomic Mobile Robots: Survey and Challenges. *Journal of Information Science & Engineering* **2017**, *33*.
18. Li, L.; Shi, D.; Jin, S.; Kang, Y.; Xue, C.; Zhou, X.; Liu, H.; Yu, X. Complete coverage problem of multiple robots with different velocities. *International Journal of Advanced Robotic Systems* **2022**, *19*, 17298806221091685.
19. Rafael Marti, G.R., Ed. *Exact and Heuristic Methods in Combinatorial Optimization*; Springer: Heidelberger Platz 3, 14197 Berlin, Germany, 2022.
20. Zhou, X.; Wang, H.; Ding, B.; Hu, T.; Shang, S. Balanced connected task allocations for multi-robot systems: an exact flow-based integer program and an approximate tree-based genetic algorithm. *Expert Systems with Applications* **2019**, *116*, 10–20.
21. Matić, D. A mixed integer linear programming model and variable neighborhood search for maximally balanced connected partition problem. *Applied Mathematics and Computation* **2014**, *237*, 85–97.
22. Sundar, K.; Rathinam, S. Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems. *Journal of Intelligent & Robotic Systems* **2017**, *88*, 513–526.
23. Chlebíková, J. Approximating the maximally balanced connected partition problem in graphs. *Information Processing Letters* **1996**, *60*, 225–230.
24. Vandermeulen, I.; Groß, R.; Kolling, A. Turn-minimizing multirobot coverage. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 1014–1020.

25. Yu, X.; Roppel, T.A.; Hung, J.Y. An optimization approach for planning robotic field coverage. In Proceedings of the IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2015, pp. 004032–004039.
26. ElGibreen, H.; Youcef-Toumi, K. Dynamic task allocation in an uncertain environment with heterogeneous multi-agents. *Autonomous Robots* **2019**, *43*, 1639–1664.
27. Gabriely, Y.; Rimon, E. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of mathematics and artificial intelligence* **2001**, *31*, 77–98.
28. Hazon, N.; Kaminka, G.A. Redundancy, efficiency and robustness in multi-robot coverage. In Proceedings of the Proceedings of the 2005 IEEE international conference on robotics and automation. IEEE, 2005, pp. 735–741.
29. Zheng, X.; Jain, S.; Koenig, S.; Kempe, D. Multi-robot forest coverage. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2005, pp. 3852–3857.
30. Zhou, X.; Wang, H.; Ding, B. How Many Robots are Enough: A Multi-Objective Genetic Algorithm for the Single-Objective Time-Limited Complete Coverage Problem. In Proceedings of the IEEE 2018 International Conference on Robotics and Automation (ICRA), Brisbane, Australia. IEEE, 2018, pp. 2380–2387.
31. Šelek, A.; Seder, M.; Brezak, M.; Petrović, I. Smooth Complete Coverage Trajectory Planning Algorithm for a Nonholonomic Robot. *Sensors* **2022**, *22*, 9269.
32. Dubins, L.E. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics* **1957**, *79*, 497–516.
33. Duckett, T.; Pearson, S.; Blackmore, S.; Grieve, B.; Chen, W.H.; Cielniak, G.; Cleaversmith, J.; Dai, J.; Davis, S.; Fox, C.; et al. Agricultural robotics: the future of robotic agriculture. *arXiv preprint arXiv:1806.06762* **2018**.
34. Xu, A.; Viriyasuthee, C.; Rekleitis, I. Optimal complete terrain coverage using an unmanned aerial vehicle. In Proceedings of the 2011 IEEE International conference on robotics and automation. IEEE, 2011, pp. 2513–2519.
35. Yu, X. Optimization approaches for a dubins vehicle in coverage planning problem and traveling salesman problems. PhD thesis, 2015.
36. Karapetyan, N.; Moulton, J.; Lewis, J.S.; Li, A.Q.; O'Kane, J.M.; Rekleitis, I. Multi-robot dubins coverage with autonomous surface vehicles. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 2373–2379.
37. Lewis, J.S.; Edwards, W.; Benson, K.; Rekleitis, I.; O'Kane, J.M. Semi-boustrophedon coverage with a dubins vehicle. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, USA. IEEE, 2017, pp. 5630–5637.
38. Lewis, J.S.; Edwards, W.; Benson, K.; Rekleitis, I.; O'Kane, J.M. Semi-boustrophedon coverage with a dubins vehicle. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, USA. IEEE, 2017, pp. 5630–5637.
39. Karapetyan, N.; Benson, K.; McKinney, C.; Taslakian, P.; Rekleitis, I. Efficient multi-robot coverage of a known environment. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, USA. IEEE, 2017, pp. 1846–1852.
40. Miller, C.E.; Tucker, A.W.; Zemlin, R.A. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)* **1960**, *7*, 326–329.
41. Bixby, B. The gurobi optimizer. *Transp. Re-search Part B* **2007**, *41*, 159–178.
42. Frieze, A.M.; Galbiati, G.; Maffioli, F. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks* **1982**, *12*, 23–39.
43. Romero, P. Simulink Drone Reference Application. https://github.com/mathworks/simulinkDroneReferenceApp/releases/tag/v2.1. 2022.