IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 20, 2020, accepted March 7, 2020, date of publication March 11, 2020, date of current version March 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980203

Path Planning for UAV to Cover Multiple Separated Convex Polygonal Regions

JUNFEI XIE[1], (Member, IEEE), LUIS RODOLFO GARCIA CARRILLO[2], (Member, IEEE), AND LEI JIN[3]

[1]Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA 92182, USA
[2]Department of Engineering, Texas A&M University–Corpus Christi, Corpus Christi, TX 78412, USA
[3]Department of Mathematics and Statistics, Texas A&M University–Corpus Christi, Corpus Christi, TX 78412, USA

Corresponding author: Junfei Xie (jxie4@sdsu.edu)

This work was supported in part by the National Science Foundation under Grant CI-1953048/1730589, in part by the San Diego State University through the University Grants Program, and in part by the Department of Defense, Network Science Division under Grant W911NF1810210.

ABSTRACT In many unmanned aerial vehicle (UAV) applications such as land assessment, search and rescue, and precision agriculture, UAVs are often required to survey multiple spatially distributed regions. To perform these applications, one of the key steps is to plan the path for the UAV to quickly cover all regions. The new path planning problem explored here, which we call the TSP-CPP problem, can be viewed as an integration of the traveling salesman problem (TSP) and the coverage path planning (CPP) problem, which has not been well studied in the literature. In this paper, we conduct a systematic investigation on the TSP-CPP problem. In particular, we first provide a mixed integer programming formulation for this new problem, and then introduce a CPP method for covering a single convex polygonal region. Based on this method, we then develop two approaches to solve the TSP-CPP problem, including 1) a dynamic programming–based exact approach that can find the (near) optimal tour, and 2) a heuristic approach that can generate high-quality tours very efficiently. Through comprehensive theoretical analyses and simulation studies, we demonstrate the optimality and efficiency of the proposed approaches.

INDEX TERMS Path planning, traveling salesman problem, coverage path planning, multiple convex polygonal regions, unmanned aerial vehicle, optimal path, heuristic algorithm.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have gained great popularity in various fields, ranging from aerospace and transportation to environmental science, geophysics, agriculture, and communication. The attracting features of UAVs including flexibility, maneuverability, low cost and high mobility make them widely applicable. Example UAV applications include environmental monitoring [1], precision agriculture [2], land assessment [3] and emergency response [4], to name a few. To realize these applications, one of the key steps is *path planning*, which aims to design the path for the UAV such that, by following this path, the UAV will successfully complete the mission while satisfying mission-specific requirements (e.g., complete the mission with the minimum time).

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

Depending on the missions, UAV path planning can be formulated as different problems, most of which fall into one of the following three categories:

1) *Shortest Path Planning Problem:* This type of problems aims to find the optimal path from the UAV's present location to a desired future location such that the travel cost (e.g., measured by travel time, travel distance or energy consumption) is minimized and the constraints (e.g., collision avoidance and mission-related constraints) are satisfied. It emerges in almost all UAV applications, and can be considered as a sub-problem of the other two types of path planning problems described below.
2) *Traveling Salesman Problem (TSP):* This type of problems seeks the optimal path that traverses a set of target locations with the minimum travel cost. Example missions include package delivery and data collection. When a fleet of UAVs with limited capacity are aimed to fulfill the demands of a set of spatially

distributed customers, the planning for these UAVs' paths, which is a variant of the TSP, is often called the Vehicle Routing Problem (VRP), or also known as the vehicle scheduling, truck dispatching or delivery problem.

3) *Coverage Path Planning (CPP) Problem:* This type of problems aims to find the optimal path that leads to full coverage of a region by the UAV's sensor footprint. Example missions include 2D and 3D mapping, land assessment, crop monitoring, etc.

In this paper, we consider a new path planning problem that seeks the optimal coverage path for multiple separated regions. This problem arises in many UAV applications. For instance, when disaster happens, multiple regions may need damage assessment. In precision agriculture, the crops that need fertilization may occupy multiple areas. In search and rescue missions, the possible active areas of the targets may be spatially distributed over different locations. Besides UAVs, many other robots applications may also encounter this problem, such as lawn mowing, room cleaning and infrastructure inspection.

This multi-regional coverage path planning problem can be considered as an integration of the TSP and CPP problems, where determining the optimal visiting order for the set of target regions can be formulated as a TSP and finding the optimal path to fully cover each region is a CPP problem. However, this integrated TSP and CPP problem, named as TSP-CPP here, is much more challenging than TSP or CPP alone, as the optimization of the region visiting order (a TSP) is interleaved with the optimization of the coverage path for each region (a CPP problem), both of which are impacted by the entrance and exit locations in each region. Therefore, to obtain the optimal multi-regional coverage path, the region visiting order, intra-regional coverage path, and the entrance and exit locations at each region should be jointly optimized and cannot be considered separately.

To the best of our knowledge, the TSP-CPP problem has not been systematically studied. In our initial investigation [5], we studied rectangular regions and introduced two exact approaches, which are proved to be optimal under mild assumptions. In this paper, we conduct a more systematic investigation on the TSP-CPP problem, with mathematical formulation provided.[1] The main contributions are summarized as follows:

- *A Dynamic Programming based Exact Approach for TSP-CPP:* We develop a dynamic programming (DP) based exact approach to solve the TSP-CPP problem, which is proved to be able to find the (near) optimal tour. This DP-based approach uses a new CPP method to generate a set of candidate paths for covering each region that are likely to yield optimal tours.
- *Optimality and Complexity Analyses:* We conduct comprehensive theoretical analyses on the optimality and complexity of the proposed CPP method and the DP-based exact approach for TSP-CPP.
- *An Efficient Heuristic Approach for TSP-CPP:* As the DP-based approach is not scalable with the number of regions, we further develop a heuristic approach that can generate high-quality tours very efficiently.
- *Extensive Simulation Studies:* We conduct extensive simulation studies to evaluate the proposed approaches from different aspects. The results demonstrate the optimality and efficiency of the proposed approaches.

In the rest of this paper, we first review the literature on path planning in Section II. We then formulate the TSP-CPP problem to be investigated mathematically in Section III. To solve the TSP-CPP problem, we first describe a CPP method for covering a single convex polygonal region in Section IV, based on which, we then introduce the DP-based exact approach for TSP-CPP in Section V and the heuristic approach in Section VI. After that, we conduct extensive simulation studies to evaluate the performance of the proposed approaches in Section VII, and finally conclude the paper in Section VIII.

II. RELATED WORK

In this section, we review typical approaches to solve the three types of path planning problems: shortest path planning problem, TSP/VRP and CPP. The current research status on the study of the TSP-CPP problem will also be discussed at the end.

A. SHORTEST PATH PLANNING PROBLEM

The typical approaches to solve the shortest path planning problem include the 1) *grid-based approaches*, such as Dijkstra's algorithm [7], A* [8], D* [9], and Phi* [10]; 2) *sampling-based approaches*, such as rapidly exploring random tree [11], probabilistic roadmap [12], and expansive space trees [13]; 3) *evolutionary algorithms*, such as genetic algorithm (GA) [14], [15], ant colony optimization [16] and particle swarm optimization [17], [18]; and 4) *optimization algorithms*, such as mixed-integer linear programming [19].

The grid-based approaches are resolution-complete, but are computationally expensive for high-dimensional complex problems [20]. The sampling-based approaches are usually more efficient and have demonstrated their advantages in solving high-dimensional problems. In addition, they are probabilistic complete as they guarantee to find a feasible solution when it exists [20], [21]. The evolutionary algorithms can also solve high-dimensional problems efficiently, but can easily be trapped in local optima. The optimization algorithms are deemed to find (near) optimal solutions. The vehicle's physical dynamics can also be incorporated to formulate path planning as an optimal control problem [22]. However, these approaches encounter the same challenge as the grid-based approaches when the problem dimension is high. To learn more about shortest path planning approaches, readers can refer to the survey articles, e.g., [20], [21], [23].

[1] This paper was presented in part at the AIAA Scitech 2019 Forum [6].

B. TSP/VRP

The TSP is a classical combinatorial optimization problem that has been well studied [24]. Typical exact methods that guarantee of finding optimal solutions include the branch-and-bound [25], branch-and-cut [26], cutting plane [27], and DP [28]. Among these methods, the branch-and-bound is convenient for TSP of 40 to 60 nodes [29]. The branch-and-cut can solve large-scale TSP of up to 2000 nodes [29]. The DP is feasible for TPS of up to 17 nodes [28]. To improve computational efficiency, non-exact methods that offer sub-optimal solutions have also been explored. Typical non-exact methods include the NN [30], 2-Opt [30], GA [31], simulated annealing [32], [33] and neural network [34]. More thorough overviews of different TSP techniques can be found in survey articles, e.g., [29], [35], [36]. A generalization of TSP to multiple salesmen, known as MTSP, is surveyed in [37]. TSP with time windows (TSPTW) is another popular variant of TSP, which restricts the salesman to visit each target location within a particular time window. Reviews of existing solutions for TSPTW can be found in [38]–[40] and the references therein.

The VRP, an extension of TSP, plays a central role in physical distributions and logistics transportation. The VRP with time window [41]–[43] and capacitated VRP [44], [45] are the two most typical variations of VRP that have been extensively studied [46]. Of interest, article [47] studies the drone delivery problem and takes the unique features of drones including limited flight range and carrying capacity into the consideration. To solve this class of problems, a wealth of exact and heuristic approaches exist (see, for example, the surveys in [37], [48], [49]), and many of them are directly derived from the procedures for solving TSP [49].

C. CPP

Based on whether full coverage can be provably achieved, the CPP methods can be classified into two categories: 1) the heuristic or randomized approaches and 2) the complete approaches [50]. The heuristic or randomized approaches are simple to implement, but take a long time to achieve full coverage. Many floor-cleaning robots [51] rely on this type of approaches. The complete approaches are advantageous in that they provide full-coverage guarantee. A two-step procedure is often adopted by this type of approaches to achieve full coverage. In particular, the first step decomposes the target region into a set of sub-regions (or called cells) by cellular decomposition. Typical cellular decomposition methods include the Trapezoidal decomposition [52], Boustrophedon decomposition [53] and Morse decomposition [54]. The second step searches for the optimal path to traverse and cover all cells. Typical search algorithms include the grid-based wavefront algorithm [55], the spiral-spanning tree coverage algorithm [56], and the backtracking spiral algorithm [57]. More comprehensive surveys of existing CPP methods can be found in [50], [58].

D. TSP-CPP

The TSP-CPP problem integrates the TSP and CPP problems, but further requires the optimization of the entrance and exit locations in each region, which impact both the region visiting order (a TSP) and intra-regional coverage paths (a CPP problem). Hence, it cannot be solved by a direct extension of existing TSP or CPP approaches, and the region visiting order, intra-regional coverage paths, and the entrance and exit locations at each region should be considered together to derive optimal paths.

Compared with other path planning problems, the TSP-CPP problem has been rarely studied in the literature. After our earlier results on TSP-CPP [5] was published in June 2018, a heuristic approach for TSP-CPP, called the two steps path planning (TSPP) was developed [59]. This approach first determines the visiting order for the regions using their centroids, and then plans the coverage path for each region. Although simple, this heuristic approach ignores the connections between region visiting order and intra-regional coverage paths, and thus produce sub-optimal paths.

III. PROBLEM FORMULATION

In this section, we describe the TSP-CPP problem to be investigated and provide a mixed integer programming formulation for this problem.

A. PROBLEM DESCRIPTION AND ASSUMPTIONS

Consider the scenario where a UAV equipped with a downward facing camera is initially located at a depot with Cartesian coordinate v_0. It is then dispatched to survey, assess or monitor a set of N separated and spatially distributed convex polygonal regions, with each region i described by a set of vertices with known coordinates $(v_{i1}, v_{i2}, \ldots, v_{im_i}) = V_i$, where v_{ij} is the coordinate of the j-th vertex of region i and m_i is the number of vertices. Note that for non-convex polygonal regions, polygon decomposition [60] can be applied to decompose the regions into a set of convex polygons, each of which can then be regarded as a target region.

Suppose there are no exclusion zones that are forbidden from visiting in the target regions. At the end of the task, the UAV is required to return back to the same depot. Suppose the UAV flies at a constant speed and at the same altitude, so the task can be described in a 2-dimensional space. The size of the camera footprint (area of ground captured in a frame) is $l \times w$ (see Fig. 1), where the length l and width w of the footprint are determined by the altitude of the UAV and features of the camera [61]. Assume that the UAV has sufficient power to complete the task. The objective of the TSP-CPP problem considered here is thus to find the optimal *tour*[2] that starts and ends at the depot, such that the UAV's camera footprints along the path will cover each target region completely and the total travel cost is minimized.

For some UAV missions, such as 3D mapping, overlapping between consecutive images is required. This can be

[2]A tour is a feasible solution to the TSP-CPP problem.

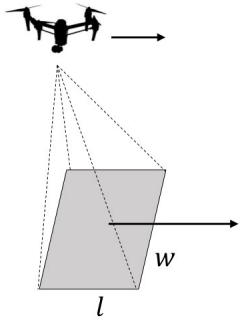

FIGURE 1. Camera footprint of the UAV.

achieved by restricting the distance between adjacent flight paths [61], [62]. Here, we consider missions that do not require image overlapping, such as crop fertilization, damage assessment, and land monitoring. However, approaches developed in this paper can be easily modified to address the needs of other survey missions.

Depending on the type of the UAV, additional constraint on the shape of the path may need to be introduced. For instance, the fixed-wing UAV cannot make abrupt directional changes, and its turn radius should be larger than a certain threshold to ensure safety.[3] This can be achieved by designing the UAV's turning curves as the Dubin's path [64], [65]. In this study, we consider multirotor UAVs that can make sharp turns with arbitrary turn radius, which will significantly increase the flexibility of the path design.

B. NOTATIONS

To formulate the TSP-CPP problem mathematically, let's first introduce a few notations. To describe the intra-regional coverage path, we let $S_i = \{s_{ip}\}$ be a set of Cartesian coordinates in region $i \in [N]$, such that by visiting these locations, the UAV will cover the whole region completely, where $p \in [n_i]$, $n_i = |S_i|$ is the cardinality of set S_i and $[n] = \{1, 2, \ldots, n\}$, $n \in \mathbb{Z}^+$. To capture the visiting order of locations in S_i, let y^i_{pq} be a decision variable such that $y^i_{pq} = 1$ if the UAV moves from location s_{ip} to location s_{iq} and $y^i_{pq} = 0$ otherwise, where $p, q \in [n_i]$. Actually p, q depend on i and are supposed to be p_i and q_i rigorously. For simplicity, we use p, q to denote p_i and q_i when there is no confusion. In the special case when region i can be fully covered by the UAV's camera footprint, $n_i = 1$ and s_{i1} is the centroid of the region.

To describe the entrance location in region i, we introduce decision variable e^i_p, such that $e^i_p = 1$ if the UAV enters region i from location s_{ip}, and $e^i_p = 0$ otherwise, where $p \in [n_i]$. Similarly, we introduce decision variable t^i_p to describe the exit location in region i, such that $t^i_p = 1$ if the UAV exits region i from location s_{ip}, and $t^i_p = 0$ otherwise. In the special case when $n_i = 1$, we have $e^i_1 = t^i_1 = 1$. Otherwise, when $n_i > 1$, we have $e^i_p + t^i_p \leq 1, \forall p \in [n_i]$, i.e., the UAV won't exit a region from the entrance location, which will lead to a longer tour.

To capture the visiting order for target regions, we let x_{ij}, $i, j \in [N] \cup \{0\}$, be a decision variable, such that $x_{ij} = 1$ if the UAV moves from region (or depot) i to region (or depot) j, and $x_{ij} = 0$ otherwise, where $i = 0$ refers to the depot.

C. MIXED INTEGER PROGRAMMING FORMULATION

With the notations defined above, a tour that originates and ends at the depot and covers all regions can be described by $S_i, y^i_{pq}, e^i_p, t^i_p, i \in [N], p, q \in [n_i]$, and $x_{ij}, i, j \in [N] \cup \{0\}$. Its total travel cost J can be calculated by

$$\begin{aligned} J = &\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\sum_{p=1}^{n_i}\sum_{q=1}^{n_j} x_{ij}t^i_p e^j_q \mathrm{d}(s_{ip}, s_{jq}) \\ &+ \sum_{i=1}^{N}\sum_{p=1}^{n_i}\sum_{q=1,q\neq p}^{n_i} y^i_{pq}\mathrm{d}(s_{ip}, s_{iq}) \\ &+ \sum_{i=1}^{N}\sum_{p=1}^{n_i} x_{0i}e^i_p\mathrm{d}(v_0, s_{ip}) + \sum_{i=1}^{N}\sum_{p=1}^{n_i} x_{i0}t^i_p\mathrm{d}(s_{ip}, v_0) \end{aligned}$$

where function $\mathrm{d}(s_{ip}, s_{jq})$ calculates the travel cost from location s_{ip} to location s_{jq}. Here we adopt the Euclidean distance to measure the travel cost. In the above cost function, the first two terms evaluate the total travel cost for inter- and intra-regional movements, respectively. The last two terms calculate the travel cost for moving from the depot to the first region in the tour and from the last region back to the depot, respectively.

To ensure that the tour is valid, the following constraints can be introduced:

$$\sum_{j=0,\, j\neq i}^{N} x_{ij} = 1, \quad \forall i \in [N] \cup \{0\} \tag{1}$$

$$\sum_{i=0,\, i\neq j}^{N} x_{ij} = 1, \quad \forall j \in [N] \cup \{0\} \tag{2}$$

$$\sum_{q=1,q\neq p}^{n_i} y^i_{pq} = 1 - t^i_p, \quad \forall i \in [N], p \in [n_i] \tag{3}$$

$$\sum_{p=1,p\neq q}^{n_i} y^i_{pq} = 1 - e^i_q, \quad \forall i \in [N], q \in [n_i] \tag{4}$$

$$\sum_{p=1}^{n_i} e^i_p = 1, \quad \sum_{p=1}^{n_i} t^i_p = 1, \quad \forall i \in [N] \tag{5}$$

$$\sum_{i,j\in M_1} x_{ij} \leq |M_1| - 1, \quad \forall M_1 \subset [N] \cup \{0\}, 2 \leq |M_1| \leq N - 1 \tag{6}$$

$$\sum_{p,q\in M_2} y^i_{pq} \leq |M_2| - 1, \quad \forall i \in [N], M_2 \subset [n_i], 2 \leq |M_2| \leq n_i - 2 \tag{7}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in [N] \cup \{0\} \tag{8}$$

$$y^i_{pq}, e^i_p, t^i_p \in \{0, 1\}, \quad \forall i \in [N], p, q \in [n_i] \tag{9}$$

[3]The minimum safe turn radius of a fixed-wing is determined by its mechanical structure [63]

$$e_p^i + t_p^i \leq 1, \quad \forall i \in [N], n_i > 1, p \in [n_i] \tag{10}$$

In the above constraints, (1)-(2) ensure each region is surveyed by the UAV exactly once. Constraints (3)-(4) ensure each location $s_{ip} \in S_i$ in region i is visited exactly once. Constraint (5) guarantees that there is only one entrance and one exit in a region and hence only one way to connect two regions. Constraints (6)-(7) eliminate sub-tours and maintain the continuity of the tour [66]. The last three constraints are introduced to ensure the decision variables take valid values.

The TSP-CPP problem can then be expressed as

$$\min_{\substack{S_i, y_{pq}^i, e_p^i, t_p^i, \forall i \in [N],\ p,q \in [n_i] \\ x_{ij}, \forall i,j \in [N] \cup \{0\}}} J$$
$$\text{subject to: constraints (1)-(10)} \tag{11}$$

Note that the TSP-CPP problem is reduced to a TSP when $n_i = 1$ for all $i \in [N]$ (each region can be fully covered by UAV's camera footprint at the centroid of the region) or a CPP problem when $N = 1$ and $n_i > 1$ (there is a single region to visit and this region is larger than the UAV's camera footprint).

D. DISCUSSIONS

The TSP-CPP problem formulated in Eq. (11) cannot be directly solved, as there are infinite number of possible choices of S_i for each region i. To address this challenge, a straightforward way is to first determine the set S_i by decomposing each region into a set of grid cells smaller than or equal to the UAV's camera footprint. The centroids of the grid cells constitute S_i, which will lead to full coverage of region i if each centroid is visited by the UAV. With S_i obtained for each region, the TSP-CPP problem formulated in Eq. (11) can then be solved using mixed integer programming solvers. Furthermore, if we relax the constraints (1), (2), (5), (6) and (10) to allow UAV to enter or exit a region for multiple times, decomposing regions into grids reduces the TSP-CPP problem to a TSP [5], which can then be solved by methods reviewed in Section II.

Although the aforementioned grid-based approach can generate optimal tours if each grid cell exactly matches the UAV's camera footprint, which can be proved based on our previous study [5], it is computationally expensive to use. In particular, its computational cost grows exponentially with the increase of the region size or the number of regions. In the following sections, we propose more efficient solutions to the TSP-CPP problem, which adopt paths of back-and-forth pattern to simplify the tour design.

IV. BACK-AND-FORTH COVERAGE PATH PLANNING FOR A SINGLE REGION

In this section, we introduce a simple CPP method that generates paths of back-and-forth pattern for covering a single convex polygonal region, as a step towards solving the TSP-CPP problem efficiently.

The back-and-forth pattern [53], [67] is widely adopted by many CPP methods, as it greatly simplifies the path design and is easy to implement. Given a convex polygonal region, the *back-and-forth path* (BFP) that covers this region can be found by a set of parallel *lines of support* (LOS), as illustrated in Fig. 2. The distance between the LOS that intersect the region at a single vertex or edge is called the *span* of the region. Here we refer to the path segments along the direction of the LOS as the *flight lines*. The *line sweep direction* is defined as the direction perpendicular to the flight lines and pointing towards the moving direction of the UAV.

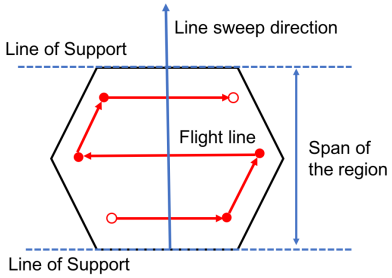

FIGURE 2. A BFP (red line) that covers a hexagonal region. The LOS are represented by dashed blue lines.

As the line sweep direction is a key factor that impacts the length of the derived BFP, in the rest of this section, we first describe how to generate the BFP given a particular line sweep direction, and then discuss how to select the line sweep direction. After that, we summarize the procedures of the proposed CPP method and end this section with theoretical analyses on the performance of the proposed method.

A. GENERATION OF BACK-AND-FORTH PATHS

Given the line sweep direction, the span of a region, denoted as W, can be determined. The minimum number of flight lines required to fully cover this region is thus $\lceil \frac{W}{w} \rceil$, where w is the width of UAV's camera footprint, indicating the maximum distance between two adjacent flight lines.[4]

To find the BFP, we first perform cellular decomposition to decompose the region into a set of sub-regions along the line sweep direction. We then construct rectangular cells that just cover sub-regions as illustrated in Fig. 3. In order to obtain steady high-resolution images, which requires the distance among flight lines to be constant, and to ensure full coverage while minimizing the path length, we set the width of the first and last cells to w and the width of intermediate cells to $\frac{w+d}{2}$, where

$$d = \frac{W - w}{\lceil W/w \rceil - 1}$$

is the distance between two adjacent flight lines. The shortest flight line that is parallel to the LOS and fully covers each cell can then be found, which has end-points $\min\{\frac{l}{2}, \frac{l'}{2}\}$ away from cell edges along the line sweep direction, where l' is the length of the cell. Note that the flight lines in the first and

[4] In case when overlapping between consecutive pictures is required in missions like 3D mapping, the minimum number of flight lines should be $\lceil \frac{W}{w(1-r)} \rceil$, where r represents the overlapping percentage.

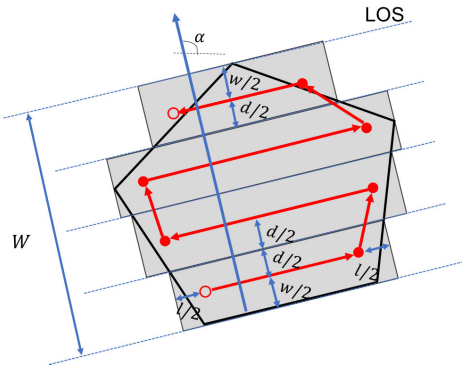

FIGURE 3. A BFP obtained at line sweep direction α.

last cells are $\frac{w}{2}$ away from the region's edges that are along the direction of LOS (see Fig. 3). In the special case when $\lceil \frac{W}{w} \rceil = 1$, only one cell of width W exists. The derived flight line is thus $\frac{W}{2}$ away from cell edges along the direction of LOS. With the flight lines, the BFP can finally be generated by connecting the flight lines in adjacent cells in such a way that the path segments in any two adjacent cells point toward opposite directions. The start and end points of the BFP are the locations from which the UAV will enter and exit the region, respectively.

B. SELECTION OF LINE SWEEP DIRECTION

It is shown in [68] that for a path of fixed length, the travel time and energy consumption will grow when the number of turns increases, as the UAV has to change its speed to make a turn. The line sweep direction that leads to the least number of turns is thus desired, which happens when the span W is minimized, as the number of turns of a BFP equals to $2(\lceil \frac{W}{w} \rceil - 1)$. According to [68], the minimum span of a convex polygonal region, defined as the *width* of the region, only appears when the line sweep direction is perpendicular to one of the region's edges.

Although we are able to find the BFP with the minimum number of turns, it may not lead to the optimal TSP-CPP tour, because the tour is not only impacted by intra-regional BFPs, but also impacted by the region visiting order and the entrance and exit locations (start and end points of the BFPs). To address this issue, we consider all BFPs that have line sweep directions perpendicular to the region's edges.

Proposition 1: Consider a convex polygonal region. Let W and L be the span of the region derived using LOS parallel to and perpendicular to an edge of the region, respectively. Then the number of possible BFPs with line sweep direction perpendicular to this edge is

$$\begin{cases} 1, & \text{if } \lceil \frac{W}{w} \rceil \lceil \frac{L}{l} \rceil = 1 \\ 2, & \text{if } \lceil \frac{W}{w} \rceil = 1 \text{ or } \lceil \frac{L}{l} \rceil = 1 \\ 4, & \text{else} \end{cases}$$

Proof: When $\lceil \frac{W}{w} \rceil \lceil \frac{L}{l} \rceil = 1$, the region can be fully covered by visiting the centroid of the region, leading to 1 possible BFP. When $\lceil \frac{W}{w} \rceil = 1$ (or $\lceil \frac{L}{l} \rceil = 1$), the region can be fully covered by a single rectangular cell with width (or length) no larger than w (or l), leading to 2 possible BFPs that traverse the same set of waypoints in reverse order. In other cases, as shown in Fig. 4, there are two possible line sweep directions that are perpendicular to the edge and two possible BFPs for each line sweep direction, thus leading to 4 possible BFPs. □

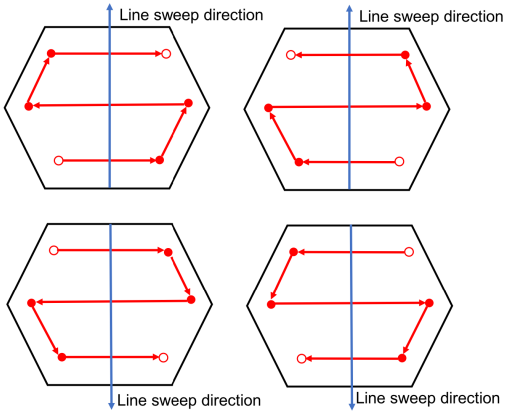

FIGURE 4. Possible BFPs with line sweep directions perpendicular to the bottom edge of a hexagonal region.

C. BACK-AND-FORTH COVERAGE PATH PLANNING ALGORITHM

Consider a convex polygonal region with its vertices $V = (v_1, v_2, \ldots, v_m)$ ordered in counter-clockwise. Algorithm 1 summarizes the procedures to generate all BFPs with line sweep directions perpendicular to the region's edges. In particular, for each edge, we first check whether parallel edge exists and has been examined (Lines 2-3). If not, we then calculate the span of the edge (Line 5) and the distance among flight lines (Lines 6-9). After that, we apply the FINDWAYPOINTS() function described in Algorithm 2 to identify the set of waypoints to be visited by the UAV (Lines 10), and finally generate all possible BFPs for this edge (Lines 11-12).

D. PERFORMANCE ANALYSIS

In this section, we analyze the performance of the back-and-forth CPP (BF-CPP) algorithm (Algorithm 1) from the aspects of full coverage guarantee, optimality, and complexity.

1) COVERAGE ANALYSIS

Lemma 1: Given a convex polygonal region and the size of UAV's camera footprint, all BFPs generated by the BF-CPP algorithm (Algorithm 1) guarantee full coverage of the region.

Proof: As each rectangular cell fully covers the corresponding sub-region and each flight line ensures full coverage of the corresponding rectangular cell, the BFPs that are derived by connecting flight lines for all sub-regions guarantee full coverage of the whole region. □

Algorithm 1 BF-CPP(V, l, w)

Input: Vertices of a region $V = (v_1, v_2, \ldots, v_m)$, size of UAV's camera footprint $l \times w$
Output: BFPs $\boldsymbol{B}$

1 **for** $i \leftarrow 1$ **to** $m-1$ **do**
2 **if** *there exists an edge* $\overline{v_j v_{j+1}}$, $j \in [i-2]$, *parallel to edge* $\overline{v_i v_{i+1}}$ **then**
3 Continue;
4 **else**
5 Find the span of edge $\overline{v_i v_{i+1}}$ by $W = \max_{j \in [m]} u_j$, where u_j is the distance between edge $\overline{v_i v_{i+1}}$ and vertex v_j;
6 **if** $\lceil \frac{W}{w} \rceil = 1$ **then**
7 $d \leftarrow 0$;
8 **else**
9 $d \leftarrow \frac{W-w}{\lceil W/w \rceil - 1}$;
10 $S \leftarrow$ FINDWAYPOINTS($W, d, \overline{v_i v_{i+1}}, l, w$);
11 Link points in S to generate all possible BFPs;
12 $\boldsymbol{B} \leftarrow \{\boldsymbol{B}, \text{BFPs}\}$;
13 **return** $\boldsymbol{B}$.

Algorithm 2 FINDWAYPOINTS($W, d, \overline{v_i v_{i+1}}, l, w$)

Input: Edge $\overline{v_i v_{i+1}}$, span W, cell width d, size of UAV's camera footprint $l \times w$
Output: Waypoints S

1 $S \leftarrow \emptyset$;
2 **if** $\lceil \frac{W}{w} \rceil > 1$ **then**
3 Perform cellular decomposition to decompose the region into $\lceil \frac{W}{w} \rceil$ sub-regions, with the width of the first and last sub-regions to be w and the width of intermediate sub-regions to be $\frac{w+d}{2}$;
4 Construct rectangular cells for each sub-region;
5 **foreach** *cell* $k \in \{1, 2, \ldots, \lceil \frac{W}{w} \rceil\}$ **do**
6 $l_k \leftarrow$ length of cell k;
7 **if** $\lceil \frac{W}{w} \rceil = 1$ **then**
8 Find the flight line that is $\frac{W}{2}$ away from cell edges parallel to $\overline{v_i v_{i+1}}$ and $\min\{\frac{l}{2}, \frac{l_k}{2}\}$ away from cell edges perpendicular to $\overline{v_i v_{i+1}}$;
9 **else if** $k = 1$ *or* $k = \lceil \frac{W}{w} \rceil$ **then**
10 Find the flight line that is $\frac{w}{2}$ away from the cell edge parallel to $\overline{v_i v_{i+1}}$ and intersecting the region at its vertex or edge, and $\min\{\frac{l}{2}, \frac{l_k}{2}\}$ away from cell edges perpendicular to $\overline{v_i v_{i+1}}$;
11 **else**
12 Find the flight line that is $\frac{d}{2}$ away from cell edges parallel to $\overline{v_i v_{i+1}}$, and $\min\{\frac{l}{2}, \frac{l_k}{2}\}$ away from cell edges perpendicular to $\overline{v_i v_{i+1}}$;
13 $S \leftarrow \{S, \text{end point(s) of the flight line}\}$;
14 **return** S.

2) OPTIMALITY ANALYSIS

Lemma 2: Given a convex polygonal region and the size of UAV's camera footprint, the BFPs generated by the BF-CPP algorithm (Algorithm 1) includes the optimal path that has the minimum number of turns.

The proof of Lemma 2 is straightforward according to the discussions in the previous subsection and is thus omitted here.

Lemma 3: Consider a rectangular region of size $L_1 \times L_2$. Given the size of UAV's camera footprint $l \times w$, the BFPs generated by the BF-CPP algorithm (Algorithm 1) includes the shortest path to cover this region, whose length is $\min\{\lceil \frac{L_1}{w} \rceil (L_2 - l) + (L_1 - w), \lceil \frac{L_2}{w} \rceil (L_1 - l) + (L_2 - w)\}$.

Proof: According to **Lemma 1** in [5], we can easily derive that the shortest travel distance for the UAV to cover the rectangular region is $\min\{\lceil \frac{L_1}{w} \rceil (L_2 - l) + (L_1 - w), \lceil \frac{L_2}{w} \rceil (L_1 - l) + (L_2 - w)\}$. This occurs when the line sweep direction of the BFP is perpendicular to the edges of length L_1 if $\lceil \frac{L_1}{w} \rceil (L_2 - l) + (L_1 - w) < \lceil \frac{L_2}{w} \rceil (L_1 - l) + (L_2 - w)$, or edges of length L_2 otherwise. □

Remark: For non-rectangular regions, the BF-CPP algorithm performs well in generating short BFPs with minimum number of turns, but cannot guarantee the shortest coverage paths.

3) COMPLEXITY ANALYSIS

Let's now analyze the computational complexity of the BF-CPP algorithm (Algorithm 1). Note that for each edge of the region, it takes $O(m)$ amount of time to compute the span W of the edge (Line 5), and $O(\lceil \frac{W}{w} \rceil)$ amount of time to find BFPs (Lines 6-12). The complexity of the BF-CPP algorithm is thus

$$O(m^2 + mW_{max}),$$

where m is the number of vertices and W_{max} is the maximum span of the region. Note that m and W_{max} characterize the shape and area of the region.

V. A DP-BASED EXACT APPROACH FOR TSP-CPP

With BF-CPP, which generates a set of candidate paths to cover a region, we are now ready to solve the TSP-CPP problem. In this section, we introduce an exact approach, which adopts the dynamic programming (DP) algorithm [69]. In the following, we first describe the key idea of the DP-based TSP-CPP approach, with pseudocode provided. We then conduct theoretical analysis on the performance of the proposed approach.

A. ALGORITHM DESCRIPTION

Suppose the UAV covers each region i by following one of the BFPs generated by the BF-CPP algorithm and we denote the set of all BFPs as $\boldsymbol{B}_i$. Let $\boldsymbol{b}_{ij}$ be the j-th possible BFP, where $j \in [|\boldsymbol{B}_i|]$ and $|\boldsymbol{B}_i|$ is the cardinality of set $\boldsymbol{B}_i$. Denote the first and last waypoints in path $\boldsymbol{b}_{ij}$ as $b_{ij}^{(1)}$ and $b_{ij}^{(e)}$, respectively, which are also the candidate locations to enter and exit region i. Now let $T \subseteq [N]$ be a non-empty subset of

target regions, and define $D(T, i, j)$, $i \in T, j \in [|\boldsymbol{B}_i|]$, as the travel cost of the optimal path that starts at the depot and fully covers each region in T, with region i being the last region in the path and $\boldsymbol{b}_{ij}$ being the j-th coverage path for region i. Under these definitions, the optimal tour to cover all regions can then be found by minimizing $D([N], i, j) + \mathrm{d}(b_{ij}^{(e)}, v_0)$, where $\mathrm{d}(b_{ij}^{(e)}, v_0)$ is the travel cost to depart from the last region i back to the depot. This new objective function allows us to find the optimal tour recursively.

To calculate the value of $D(T, i, j)$, we consider the path that ends at the region visited prior to region i. In particular, suppose the UAV visits region $k \in T$ by following path $\boldsymbol{b}_{kh} \in \boldsymbol{B}_k$ prior to region i. Therefore, $D(T, i, j)$ can be computed by adding three parts: 1) $D(T \setminus \{i\}, k, h)$, 2) the cost to travel from region k to region i, i.e., $\mathrm{d}(b_{kh}^{(e)}, b_{ij}^{(1)})$, and 3) the travel cost of path $\boldsymbol{b}_{ij}$. As k can be any region in set $T \setminus \{i\}$ and $\boldsymbol{b}_{kh}$ can be any path in set $\boldsymbol{B}_k$, $D(T, i, j)$ is solved by considering all $k \in T \setminus \{i\}$ and all $h \in [|\boldsymbol{B}_k|]$ for each k. The mathematical formulation of $D(T, i, j)$ is then given by

$$D(T, i, j) = \begin{cases} \infty, & \text{if } i \notin T \\ \mathrm{d}(v_0, b_{ij}^{(1)}) + \mathrm{g}(\boldsymbol{b}_{ij}), & \text{if } T = \{i\} \\ \min\limits_{\substack{k \in T \setminus \{i\} \\ h \in [|\boldsymbol{B}_k|]}} D(T \setminus \{i\}, k, h) & \\ \quad + \mathrm{d}(b_{kh}^{(e)}, b_{ij}^{(1)}) + \mathrm{g}(\boldsymbol{b}_{ij}), & \text{else} \end{cases} \tag{12}$$

where function $\mathrm{g}(\boldsymbol{b}_{ij})$ computes the travel cost of path $\boldsymbol{b}_{ij}$.

Algorithm 3 provides the pseudocode of the proposed DP-based exact approach for TSP-CPP. In particular, the BFPs for each region are first generated using BF-CPP (Lines 1-5). DP is then performed to search for the optimal tour (Lines 6-17). Table P in Lines 5 and 15 is used to track the locations along the path.

Algorithm 3 EXACT(v_0, $\boldsymbol{V}$, l, w)

Input: Depot v_0, vertices of all regions $\boldsymbol{V} = (V_1, V_2, \ldots, V_N)$, size of UAV's camera footprint $l \times w$.
Output: Tour τ^*, travel cost J^*.

1 **for** $i \leftarrow 1$ **to** N **do**
2 $\boldsymbol{B}_i \leftarrow$ BF-CPP(V_i, l, w);
3 **for** $j \leftarrow 1$ **to** $|\boldsymbol{B}_i|$ **do**
4 $D(\{i\}, i, j) \leftarrow \mathrm{d}(v_0, b_{ij}^{(1)}) + \mathrm{g}(\boldsymbol{b}_{ij})$;
5 $P(\{i\}, i, j) \leftarrow (v_0, \boldsymbol{b}_{ij})$;
6 **for** $t \leftarrow 2$ **to** N **do**
7 **foreach** $T \subseteq [N]$ *where* $|T| = t$ **do**
8 **foreach** $i \in T$ **do**
9 **for** $j \leftarrow 1$ **to** $|\boldsymbol{B}_i|$ **do**
10 **foreach** $k \in T - \{i\}$ **do**
11 **for** $h \leftarrow 1$ **to** $|\boldsymbol{B}_k|$ **do**
12 $dist \leftarrow D(T \setminus \{i\}, k, h) + \mathrm{d}(b_{kh}^{(e)}, b_{ij}^{(1)}) + \mathrm{g}(\boldsymbol{b}_{ij})$;
13 **if** $dist < D(T, i, j)$ **then**
14 $D(T, i, j) \leftarrow dist$;
15 $P(T, i, j) \leftarrow (\boldsymbol{b}_{kh}, \boldsymbol{b}_{ij})$;
16 $J^* \leftarrow \min\limits_{i \in [N], j \in [|\boldsymbol{B}_i|]} D([N], i, j) + \mathrm{d}(v_0, b_{ij}^{(e)})$;
17 Backtrack over arcs in table P to derive the optimal tour τ^*;
18 **return** τ^*, J^*.

B. PERFORMANCE ANALYSIS

In this section, we first prove in Theorem 1 the optimality of the proposed DP-based exact approach and then analyze its computational complexity.

1) OPTIMALITY ANALYSIS

Theorem 1: Consider the TSP-CPP problem formulated in Eq. (11) *and assume that the UAV can only choose from the BFPs generated by the BF-CPP algorithm (Algorithm 1) to cover each region. The tour found by the DP-based exact approach (Algorithm 3) is optimal.*

Proof: The optimality of the DP-based approach can be proved by induction [28]. In particular, for any collection of regions T, as $D(T, i, j)$ calculated in each recursion step is the travel cost of the optimal path that fully covers all regions in T, and ends at region i with $\boldsymbol{b}_{ij}$ as the path adopted for covering region i, Lines 16-17 in Algorithm 3 find the optimal tour with the minimum travel cost to cover all regions. □

2) COMPLEXITY ANALYSIS

Let's now analyze the computational complexity of the proposed DP-based exact approach for TSP-CPP (Algorithm 3). Let m_i and W_i^{max} be the number of edges and the maximum span of region i, respectively. As the complexity of Algorithm 1 is $O\left(m_i^2 + m_i W_i^{max}\right)$ for each region i, it takes $O\left(\sum_{i=1}^{N} m_i^2 + m_i W_i^{max}\right)$ amount of time to process Lines 1-5.

Furthermore, it can be derived that it takes $O\Big(\sum_{t=2}^{N} \sum_{T \subseteq [N], |T|=t} \sum_{i \in T} \sum_{k \in T \setminus \{i\}} |\boldsymbol{B}_i||\boldsymbol{B}_k|\Big)$ amount of time to process Lines 6-15. To simplify this expression, let $\hat{m} = \max\limits_{i \in [N]} m_i$ and $\hat{W} = \max\limits_{i \in [N]} W_i^{max}$. As $|\boldsymbol{B}_i| \leq 4m_i \leq 4\hat{m}$ according to Proposition 1, the complexity of Lines 6-15 can be simplified to $O\left(16\hat{m}^2 \sum_{t=2}^{N} t(t-1)\binom{N}{t}\right) = O\left(4\hat{m}^2(N^2 - N)2^N\right) = O\left(\hat{m}^2 N^2 2^N\right)$. As the minimum cost in Line 16 can be computed in $O\left(\sum_{i=1}^{N} \log |\boldsymbol{B}_i|\right) = O\left(N \log \hat{m}\right)$ amount of time, the complexity of the DP-based approach is thus $O(\sum_{i=1}^{N}(m_i^2 + m_i W_i^{max}) + \hat{m}^2 N^2 2^N + N \log \hat{m})$, which can further simplified to

$$O\left(\hat{m} N^2 \hat{W} + \hat{m}^2 N^2 2^N\right).$$

This indicates that the DP-based approach scales well with respect to the region size, characterized by $\hat{m}$ and $\hat{W}$,

Algorithm 4 HEURISTIC(v_0, $\boldsymbol{V}$, l, w)

Input: Depot v_0, vertices of all regions $\boldsymbol{V}$, size of UAV's camera footprint $l \times w$
Output: Tour $\boldsymbol{\tau}^*$, travel cost J^*

1 **for** $i \leftarrow 1$ **to** N **do**
2 $c_i \leftarrow$ centroid of region i;
3 $\boldsymbol{B}_i \leftarrow$ BF-CPP(V_i, l, w);
4 $c_0 \leftarrow v_0$, $\boldsymbol{c} \leftarrow (c_0, c_1, \ldots, c_m)$, $\boldsymbol{\mathcal{B}} \leftarrow (\boldsymbol{B}_1, \ldots, \boldsymbol{B}_m)$;
5 $(\boldsymbol{\tau}, \boldsymbol{o}) \leftarrow$ INITIALIZATION($\boldsymbol{c}$, $\boldsymbol{\mathcal{B}}$);
6 $\boldsymbol{\tau}^* \leftarrow$ IMPROVETOUR($\boldsymbol{\tau}$, $\boldsymbol{o}$, $\boldsymbol{c}$, $\boldsymbol{\mathcal{B}}$);
7 $J^* \leftarrow \mathrm{g}(\boldsymbol{\tau}^*)$;
8 **return** $\boldsymbol{\tau}^*$, J^*.

but suffers from the curse of dimensionality with respect to the number of regions, N.

VI. A HEURISTIC APPROACH FOR TSP-CPP

While the DP-based exact approach generates (near) optimal tours, it is not scalable with the number of regions. To address large-scale TSP-CPP problems, more efficient approaches are desired. In this section, we introduce an innovative heuristic approach for TSP-CPP, which generates high-quality tours very efficiently.

Our heuristic approach solves the TSP-CPP problem by performing two steps: 1) *initialization* that initializes the tour and 2) *tour improvement* that iteratively improves the tour until convergence. The procedures of the proposed approach are summarized in Algorithm 4. Next, let's describe each step in more detail.

A. INITIALIZATION

The first step aims to quickly initialize the tour. A straightforward way is to generate a random tour, which however may cause the convergence of the tour improvement step to be very slow. Another approach is to apply the two steps path planning (TSPP) algorithm [59], which first uses genetic algorithm (GA) [31] to find the visiting order for the regions, with each region regarded as a point located at the centroid of the region. It then finds the coverage path of back-and-forth pattern for each region, with the start and end locations being the centroids of the regions to be visited right before and right after the current region, respectively. Although the TSPP is simple to implement and performs well for small number of regions, it may take a long time to generate a good tour when the number of regions is large.

To achieve higher efficiency, we propose a modified nearest neighbor (NN) algorithm with pseudocode provided in Algorithm 5. In this algorithm, the NN [70] is first applied to determine the region visiting order based on regions' centroids, which finds NN paths for multiple starting points and returns the best of those paths (Lines 1-11). After that, the complete tour to fully cover all regions is determined using function FINDTOUR() described in Algorithm 6 (Line 12), which selects the best BFP for each

Algorithm 5 INITIALIZATION($\boldsymbol{c}$, $\boldsymbol{\mathcal{B}}$)

Input: Region centroid $\boldsymbol{c}$, BFPs $\boldsymbol{\mathcal{B}}$
Output: Tour $\boldsymbol{\tau}$, region visiting order $\boldsymbol{o}^*$

1 **for** $i \leftarrow 0$ **to** N **do**
2 $\boldsymbol{o}_i(0) \leftarrow i$;
3 $h \leftarrow i$;
4 $\mathcal{W} \leftarrow \{0, 1, 2, \ldots, N\} \setminus \{h\}$;
5 **for** $k \leftarrow 1$ **to** N **do**
6 $j \leftarrow \arg\min_{t \in \mathcal{W}} \mathrm{d}(c_h, c_t)$;
7 $\boldsymbol{o}_i(k) \leftarrow j$;
8 $\mathcal{W} \leftarrow \mathcal{W} \setminus \{j\}$;
9 $h \leftarrow j$;
10 $\boldsymbol{o}^* \leftarrow \arg\min_{i \in \{0,1,\ldots,N\}} \mathrm{g}(\boldsymbol{o}_i)$;
11 $\boldsymbol{\tau} \leftarrow$ FINDTOUR($\boldsymbol{o}^*$, $\boldsymbol{c}$, $\boldsymbol{\mathcal{B}}$);
12 **return** $\boldsymbol{\tau}$, $\boldsymbol{o}^*$.

Algorithm 6 FINDTOUR($\boldsymbol{o}$, $\boldsymbol{c}$, $\boldsymbol{\mathcal{B}}$)

Input: Region visiting order $\boldsymbol{o}$, region centroids $\boldsymbol{c}$, BFPs $\boldsymbol{\mathcal{B}}$
Output: Tour $\boldsymbol{\tau}$

1 Sort $\boldsymbol{o}$ without changing the region visiting order to make $\boldsymbol{o}(0) = 0$;
2 $\boldsymbol{\tau} \leftarrow \{v_0\}$;
3 **for** $t \leftarrow 1$ **to** N **do**
4 $i \leftarrow \boldsymbol{o}(t)$;
5 **if** $t < N$ **then**
6 $k \leftarrow \boldsymbol{o}(t+1)$;
7 **else**
8 $k \leftarrow 0$;
9 $\boldsymbol{b}_{ij} \leftarrow$ $\arg\min_{h \in [|\boldsymbol{B}_i|]} g(\boldsymbol{b}_{ih}) + \mathrm{d}(\boldsymbol{\tau}(\mathrm{end}), b_{ih}^{(1)}) + \mathrm{d}(b_{ih}^{(e)}, c_k)$, where $\boldsymbol{\tau}(\mathrm{end})$ is the last location in path $\boldsymbol{\tau}$;
10 $\boldsymbol{\tau} \leftarrow \{\boldsymbol{\tau}, \boldsymbol{b}_{ij}\}$;
11 $\boldsymbol{\tau} \leftarrow \{\boldsymbol{\tau}, v_0\}$;
12 **return** $\boldsymbol{\tau}$.

region sequentially in the region visiting order, with the start location being the depot and end location being the centroid of the region to be visited next.

B. TOUR IMPROVEMENT

As the initialization step ignores the correlation between region visiting order and intra-regional paths, the generated tour may not be optimal. This issue is especially outstanding when the problem scale is large. The tour improvement step aims to resolve this issue to some extent by using a modified 2-Opt algorithm [30], [71] to improve the tour in an iterative manner. In particular, at each iteration, we perform a 2-Opt move to update the region visiting order, which replaces two region-to-region links with two other links

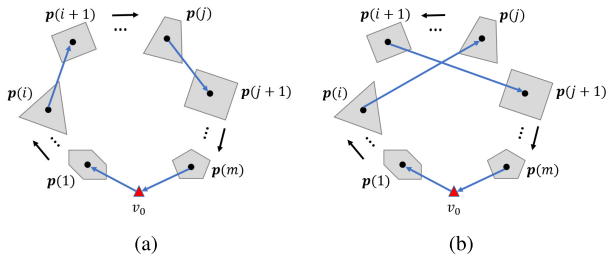

FIGURE 5. Region visiting order a) before and b) after applying a 2-Opt move.

Algorithm 7 IMPROVETOUR($\boldsymbol{\tau}, \boldsymbol{o}, \boldsymbol{c}, \mathcal{B}$)

Input: Tour $\boldsymbol{\tau}$, region visiting order $\boldsymbol{o}$, region centroids $\boldsymbol{c}$, BFPs $\mathcal{B}$
Output: Improved tour $\boldsymbol{\tau}^*$

1 $\boldsymbol{\tau}^* \leftarrow \boldsymbol{\tau}$;
2 $cost \leftarrow \infty$;
3 **while** $g(\boldsymbol{\tau}^*) < cost$ **do**
4 $cost \leftarrow g(\boldsymbol{\tau}^*)$;
5 **for** $i \leftarrow 0$ **to** $N-1$ **do**
6 **for** $j \leftarrow i+2$ **to** $N+1$ **do**
7 **if** $d(c_{\boldsymbol{o}(i)}, c_{\boldsymbol{o}(i+1)}) > d(c_{\boldsymbol{o}(i)}, c_{\boldsymbol{o}(j)})$ **then**
8 $\boldsymbol{o}' \leftarrow \boldsymbol{o}$ with links $(\boldsymbol{o}(i), \boldsymbol{o}(i+1))$ and $(\boldsymbol{o}(j), \boldsymbol{o}(j+1))$ replaced with $(\boldsymbol{o}(i), \boldsymbol{o}(j))$ and $(\boldsymbol{o}(i+1), \boldsymbol{o}(j+1))$, respectively;
9 $\boldsymbol{\tau}' \leftarrow$ FINDTOUR($\boldsymbol{o}', \boldsymbol{c}, \mathcal{B}$);
10 **if** $g(\boldsymbol{\tau}') < g(\boldsymbol{\tau})$ **then**
11 $\boldsymbol{\tau}^* \leftarrow \boldsymbol{\tau}'$;
12 **return** $\boldsymbol{\tau}^*$.

(see Fig. 5 for an illustration). With the new region visiting order, we then use the FINDTOUR() function (Algorithm 6) to find the corresponding tour, which is then used to update the current tour if shorter in length. This procedure is repeated until no more improvements are possible.

The original 2-Opt procedure evaluates all possible moves for one that shortens the tour, which can make the search very time consuming, especially when the problem scale is large or the initial tour is far from optimal. To improve the efficiency, we adopt a constraint proposed in [30] to reduce the search space. Algorithm 7 provides the pseudocode of the tour improvement step, where the constraint in Line 7 limits the 2-Opt moves to be evaluated. It is worthy to note that this constraint won't exclude an improving move for TSP [30], but this may not be the case for TSP-CPP, due to the influence of intra-regional paths. Nevertheless, as we will show in the simulation studies, this constraint significantly speeds up the computation and only degrades the optimality slightly. For convenience of reference, we name this NN- and 2-Opt-based heuristic approach with constraint as Fast NN-2Opt, and the one without constraint as NN-2Opt.

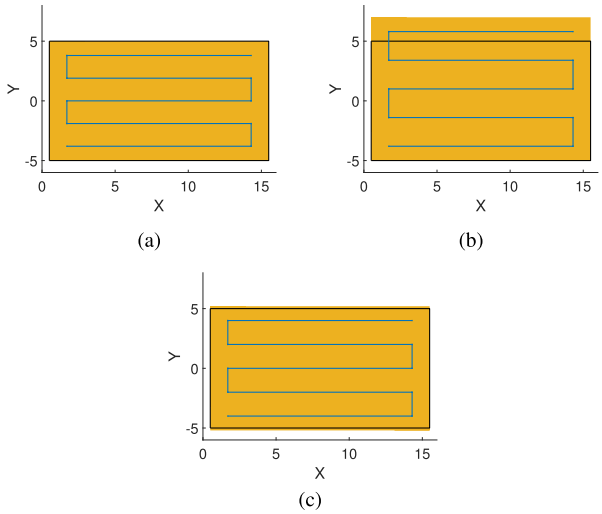

FIGURE 6. The BFPs (blue lines) generated by a) the proposed BF-CPP, b) benchmark method 1, and c) benchmark method 2 to cover a rectangular region. The area covered by the UAV is highlighted in orange.

VII. SIMULATION STUDIES

In this section, we conduct simulation studies to evaluate the performance of the proposed approaches, including the BF-CPP algorithm, DP-based exact approach for TSP-CPP and the heuristic approach for TSP-CPP. All approaches are implemented using MATLAB, and the experiments are conducted on a Microsoft Surface Pro 7 with Intel Core i5, 8GB memory and 256GB storage.

A. PERFORMANCE OF THE BF-CPP ALGORITHM

Lemmas 1 and 3 indicate that the BFPs generated by the BF-CPP algorithm guarantee the full coverage with the shortest tour in case of rectangular regions. In this study, we design experiments to illustrate the capability of BF-CPP in dealing with rectangular regions and also non-rectangular regions.

1) BENCHMARK CPP METHODS

For comparison, we also implement two benchmark methods that adopt traditional cellular decomposition methods.

- **Benchmark Method 1:** This method decomposes a region into cells of the same width equal to $d = w$. It is adopted in TSPP [59].
- **Benchmark Method 2:** This method decomposes a region into cells of the same width equal to $d = \frac{W}{\lceil W/w \rceil}$.

For both benchmark methods, flight lines that are $\frac{d}{2}$ away from cell edges along the direction of LOS and $\min\{\frac{l}{2}, \frac{l'}{2}\}$ away from cell edges along the line sweep direction are found to cover a given region, where l' is the cell length.

2) COVERAGE PATH PLANNING FOR A RECTANGULAR REGION

In the first experiment, we consider a rectangular region of size 15×10, with the size of UAV's camera footprint set to 2.4×2.4. Fig. 6 shows the BFPs with upwards line sweep

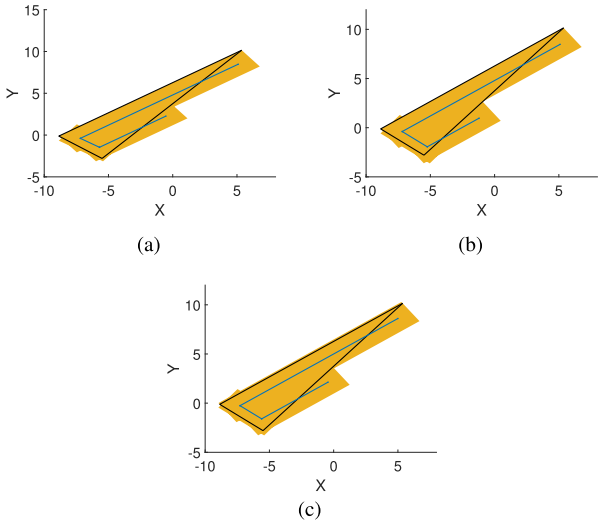

FIGURE 7. The BFPs found by a) the proposed BF-CPP; b) benchmark method 1; and c) benchmark method 2 to cover a triangular region.

direction generated by the three CPP methods. As shown in the figure, all three paths lead to complete coverage of the region. Moreover, the path generated by our method (Fig. 6(a)) has the shortest length of 70.6, leading to the smallest covered area that exactly matches the region. The first benchmark method (Fig. 6(b)) generates the longest path of length 72.6, leading to the largest covered area. Of note, in Fig. 6(b), the top flight line is outside of the region, as the others are not sufficient to fully cover the region. For the second benchmark method (Fig. 6(c)), it generates a path of length 71 and has the covered area slightly larger than the region.

3) COVERAGE PATH PLANNING FOR A NON-RECTANGULAR REGION

As we mentioned before, for non-rectangular regions, our BF-CPP algorithm may not be able to find the shortest BFPs. To demonstrate this, we consider a triangular region of size $4.3 \times 16.9 \times 17.6$ in this experiment. The size of UAV's camera footprint is still set to 2.4×2.4. The BFPs with downwards line sweep direction generated by the three CPP methods are shown in Fig. 7. Among them, the path generated by the first benchmark method (Fig. 7(b)) has the shortest length of 22.6802, and the one generated by the second benchmark method (Fig. 7(c)) is the longest with length of 23.6854. The length of the path generated by our method is 23.3703.

By comparing Fig. 7(b) and Fig. 7(a), we can find that, the first benchmark method generates a shorter path because its cells have a larger width than the ones defined in our method, which cause the flight line in the small cell in Fig. 7(b) much shorter than the one in Fig. 7(a), thus leading a shorter BFP. Of note, this situation does not happen when the region is a rectangle, as the cell width does not impact the length of the flight lines for rectangular regions. On the contrary, a larger cell width will lead to a larger distance between adjacent flight lines and thus longer coverage

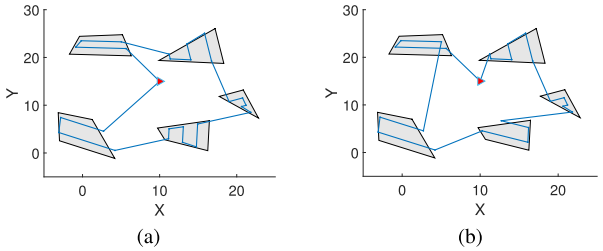

FIGURE 8. The tours generated by the a) proposed DP-based approach and b) TSPP. Shaded grey areas and the red triangle represent regions and the depot, respectively.

paths for rectangular regions. This further explains the optimality of our method in case of rectangular regions. Moreover, it is also worthy to note that although our method does not guarantee the shortest BFP for non-rectangular regions, it generates shorter tours, compared with benchmark methods, in most cases we have evaluated.

B. PERFORMANCE OF THE DP-BASED EXACT APPROACH

In this study, we investigate the performance of the DP-based exact approach for TSP-CPP in terms of optimality and efficiency.

1) OPTIMALITY STUDY

Theorem 1 points out that our DP-based approach can find the optimal tour given that the UAV follows the BFPs generated by the BF-CPP algorithm to cover the regions. In this study, we design two experiments to demonstrate the optimality of DP-based approach when its underlying assumption is satisfied and also when the assumption is violated.

Experiment 1: The first experiment aims to demonstrate the performance of DP-based approach when its underlying assumption is satisfied. In this experiment, we consider five convex polygonal regions distributed randomly around the depot. The size of UAV's camera footprint is set to 1.5×3. For comparison, we also implement the TSPP [59], in which we use DP to determine the region visiting order [28]. For fairness, we also let the TSPP adopt BFPs generated by our BF-CPP algorithm.

Fig. 8 visualizes the tours generated by the DP-based approach and the TSPP, whose lengths are 117.9461 and 124.2275, respectively. The TSPP generates a longer tour, because it considers the optimization of the region visiting order and the optimization of intra-regional paths separately. Of note, we also evaluated our heuristic approach. In particular, the NN-2Opt finds the same tour as the one generated by the DP-based approach, and the Fast NN-2Opt generates a slightly longer tour of length 123.207. More investigations on the proposed heuristic approach will be discussed in Section VII-C.

Experiment 2: The second experiment aims to demonstrate the performance of DP-based approach when its underlying assumption is violated. To relax this assumption, we allow the UAV to follow non-BFPs by adopting the grid-based approach described in Section III-D, which decomposes each

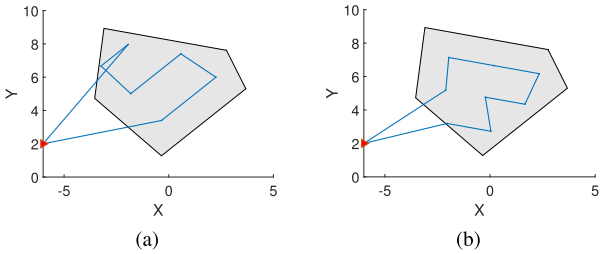

FIGURE 9. The tours generated a) before and b) after its underlying assumption is relaxed.

region into grids smaller than or equal to the UAV's camera footprint, and generates coverage paths that traverse the centroids of all the grids. We then consider a single pentagonal region and set the size of UAV's camera footprint to 3×3.

Fig. 9 shows the tours generated by the DP-based approach before and after its underlying assumption is relaxed. As expected, when UAV is not restricted to follow BFPs, a shorter tour is generated, which has a length of 23.6047. In contrast, as the BFPs constrain the entrance and exit locations in a region, the resulting tour is longer, which has a length of 26.3118. Furthermore, we note that the intra-regional path of back-and-forth pattern (Fig. 4(a)), which has a length of 13.2677, is shorter than the one generated by the grid-based approach (Fig. 4(b)), which has a length of 14.4361. This demonstrates the significant role of entrance and exit locations in determining the tour. Finally, we would like to mention again that although the grid-based approach can generate shorter tours, it is neither scalable with the region size nor the number of regions, as we have illustrated in [5].

2) EFFICIENCY STUDY

In Section V-B, we have derived the complexity of the DP-based approach, which is $O\left(\hat{m}N^2\hat{W} + \hat{m}^2N^2 2^N\right)$. In this study, we design three experiments to further validate this result and demonstrate the impact of the three parameters $(N, \hat{m}, \hat{W})$ on the efficiency of the DP-based approach. For comparison, we also evaluate the TSPP and the proposed heuristic approach, Fast NN-2Opt.

Experiment 1: In the first experiment, we study the impact of the number of regions N. To generate regions of constant size, we apply the `nsidedpoly()` function in MATLAB, which produces regular polygons. For each region, we fix the number of edges and the length of each edge to 5 and 128, respectively. We then generate different numbers of regions, and execute the three compared approaches to measure their execution times for each scenario. The size of UAV's camera footprint is set to 1.5×3. To reduce uncertainty, each experiment is repeated for 10 times and the mean execution times are recorded. Fig. 10(a) shows the simulation results. As expected, the execution time of the DP-based approach grows exponentially with the increase of the number of regions, and both TSPP and Fast NN-2Opt are significantly more efficient than DP-based approach.

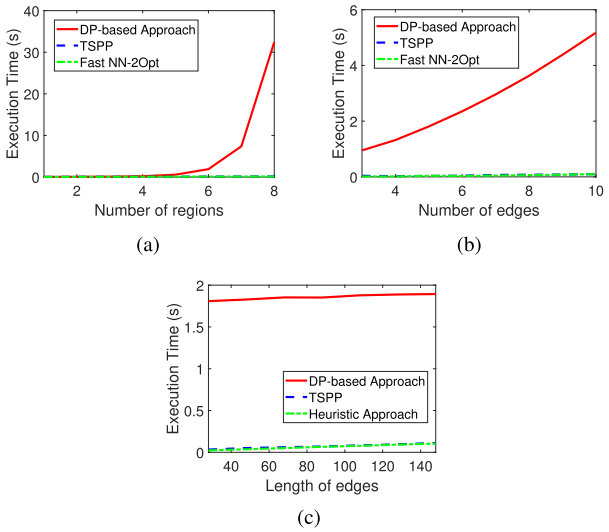

FIGURE 10. The mean execution times of different approaches with the increase of a) the number of regions, b) the number of edges of each region, and c) the length of the edges of each region.

Experiment 2: In the second experiment, we study the impact of the number of edges $\hat{m}$ of a region. In particular, we fix the number of regions and the length of the edges of each region to 6 and 28, respectively, and set the size of UAV's camera footprint to 1.5×3. We then increase the number of edges of each region simultaneously from 3 to 10. As shown in Fig. 10(b), the execution time of the DP-based approach grows quadratically with the increase of the number of edges, and both TSPP and Fast NN-2Opt outperform DP-based approach.

Experiment 3: In the third experiment, we study the impact of the maximum span $\hat{W}$ of a region. This is achieved by varying the length of the edges of each regular polygonal region. The number of regions and the number of edges of each region are fixed to 6 and 5, respectively. The size of UAV's camera footprint is still set to 1.5×3. As shown in Fig. 7(c), the execution time of the DP-based approach grows linearly with the increase of the length of edges, and both TSPP and Fast NN-2Opt are more efficient than DP-based approach.

C. PERFORMANCE OF THE HEURISTIC APPROACH

In this study, we investigate the performance of the proposed heuristic approach for TSP-CPP in terms of optimality and efficiency.

1) EXPERIMENT SETTINGS

For a thorough understanding, we compare the following six TSP-CPP methods in this study:

- **TSPP [59]:** This method first uses GA to find the region visiting order, and then determines the intra-regional paths and the tour.
- **Fast NN-2Opt:** This is the proposed method described in Algorithm 4.

- **Fast GA-2Opt:** This is an alternative method to the Fast NN-2Opt, which uses GA, instead of NN, to find the region visiting order in the initialization step.
- **NN-2Opt:** This is the proposed method described in Algorithm 4 without introducing the constraint in Line 7 of Algorithm 7.
- **GA-2Opt:** This is an alternative method to the NN-2Opt, which uses GA, instead of NN, to find the region visiting order in the initialization step.
- **Extended GA:** This method applies GA to optimize the tour, which defines the chromosome as an ordered list of regions presenting the region visiting order and defines the fitness function as the travel cost of the resulting tour derived using the FINDTOUR() function in Algorithm 6.

To evaluate the performance of these TSP-CPP methods, we vary the number of regions and consider three scenarios: $N = 20$, $N = 50$ and $N = 100$. The regions are randomly generated convex polygons, which do not overlap between with each other.

As the performance of GA largely depends on its parameters, we also vary its parameter setting. In this study, we implement the GA based on the code in [72], which have two parameters to configure: 1) number of iterations, denoted as $\mathcal{I}$, and 2) population size, denoted as ρ. To enable early termination, we further introduce a parameter κ, such that the algorithm will terminate either when the best fitness score is unchanged for κ successive generations or when $\mathcal{I}$ iterations have been evaluated. We then consider the settings with three parameters:

- **Setting 1:** $\mathcal{I} = 10^4$, $\kappa = 10$, and $\rho = 100$
- **Setting 2:** $\mathcal{I} = 10^5$, $\kappa = 50$, and $\rho = 100$
- **Setting 3:** $\mathcal{I} = 10^6$, $\kappa = 100$, and $\rho = 100$

These settings will impact the performance of four methods that use GA, including TSPP, Fast GA-2Opt, GA-2Opt and Extended GA.

In all experiments, we set the size of UAV's camera footprint to 1.5×3. Each experiment is repeated for 10 times, and the mean execution time of each method and the mean travel cost of each derived tour, measured by the Euclidean distance, are recorded.

2) OPTIMALITY STUDY

Fig. 11 shows the mean travel cost of the tour generated by each TSP-CPP method, grouped by GA's parameter setting, when different numbers of regions are present. As an illustration, we also selectively show in Fig. 12 the tour generated by each method when $N = 100$ and GA adopts the first parameter setting.

From Fig. 11, we first see that the proposed NN-2Opt achieves the best performance in most scenarios. Fast NN-2Opt is less optimal than NN-2Opt, as the additional constraint it adopts can exclude good 2-Opt moves. Nevertheless, such performance degradation is not significant, as shown in Fig. 11 and Fig. 12. Note that, as GA is not used in Fast NN-2Opt and NN-2Opt, the tours generated by these two methods are deterministic and are not influenced by GA's parameters.

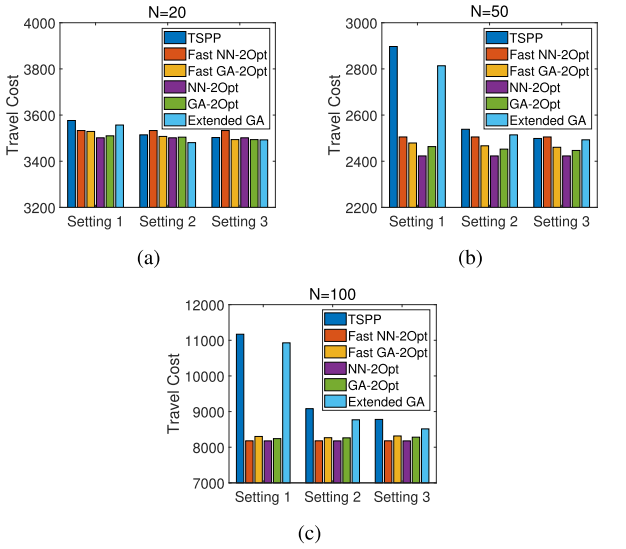

FIGURE 11. The mean travel costs of the tours generated by different approaches when there are a) $N = 20$, b) $N = 50$, and c) $N = 100$ regions.

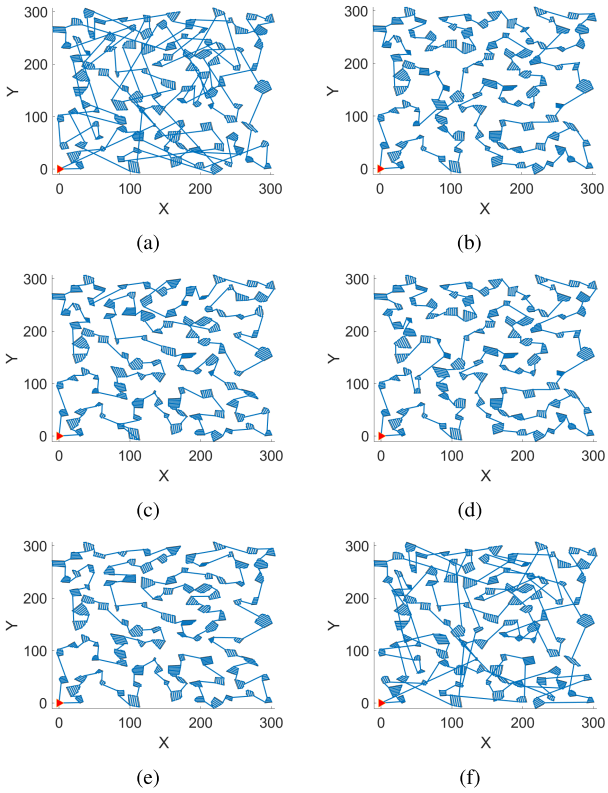

FIGURE 12. The tour generated by a) TSPP, b) Fast NN-2Opt, c) Fast GA-2Opt, d) NN-2Opt, e) GA-2Opt, and f) Extended GA when there are $N = 100$ regions and Setting 1 is adopted to configure the GA.

We can also see from Fig. 11 that TSPP generates the longest tour in most scenarios, especially when the number of regions is large. Increasing $\mathcal{I}$ and κ enhances the performance of TSPP, due to the improved quality of the GA solution (region visiting order). Moreover, we can observe from both

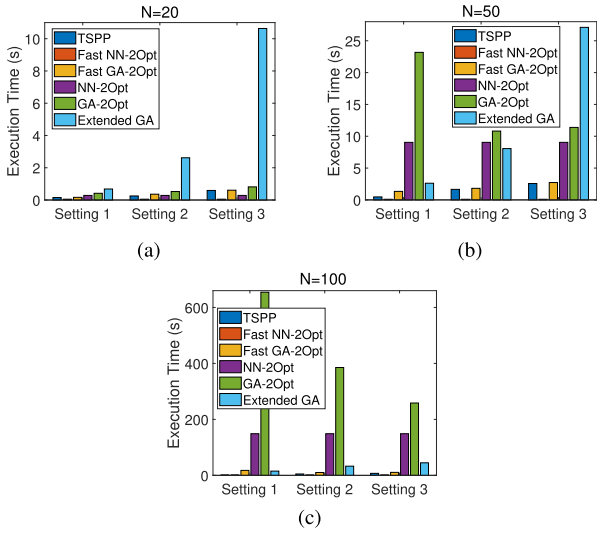

FIGURE 13. The mean execution times of different TSP-CPP methods when there are a) $N = 20$, b) $N = 50$ and c) $N = 100$ regions.

Fig. 11 and Fig. 12 that GA-2Opt and Fast GA-2Opt achieve comparable performance with NN-2Opt and Fast NN-2Opt, and always outperform TSPP. This demonstrates the effectiveness of the proposed tour improvement step in improving the quality of the tour. In addition, GA-2Opt outperforms Fast GA-2Opt, as it does not exclude any possible 2-Opt moves. We can also observe that the Extended GA outperforms TSPP in all scenarios, as it considers the inter-dependencies of the region visiting order and the intra-regional paths. In addition, the performance of Extended GA improves with $\mathcal{I}$ and κ, and sometimes even generates the shortest tours.

3) EFFICIENCY STUDY

Fig. 13 shows the mean execution times of different TSP-CPP methods. As we can see from the figure, the Fast NN-2Opt outperforms the other methods in all scenarios. NN-2Opt is less efficient than Fast NN-2Opt, as it evaluates all possible 2-Opt moves. Moreover, its efficiency degrades quickly when the number of regions N increases. This is because, with the increase of N, the number of 2-Opt moves required to find a shorter tour grows quadratically, according to Algorithm 7, and the number of possible improvements grows quickly as well.

We can also see from Fig. 13 that the TSPP is the second most efficient method. The Fast GA-2Opt is relatively efficient, but is less efficient than TSPP, as it further conducts tour improvement after an initial tour is obtained. The GA-2Opt is less efficient than Fast GA-2Opt, as expected, and its efficiency degrades significantly when the number of regions increases. The Extended GA is the least efficient when the number of regions is small, but outperforms NN-2Opt and GA-2Opt when the number of regions is large.

Another observation from Fig. 13 is that the efficiency of both Extended GA and TSPP degrades with the increase of $\mathcal{I}$ and κ. Nevertheless, when the number of regions is large, the efficiency of Fast GA-2Opt and GA-2Opt improves with the increase of $\mathcal{I}$ and κ. This is because, in Fast GA-2Opt and GA-2Opt, the quality of the initial tour generated by the GA-based method improves with the increase of $\mathcal{I}$ and κ, which causes the tour improvement step to converge more quickly. Though the time reduced in the tour improvement step is not obvious for small number of regions, in which case the initialization step consumes the most time, it becomes evident when the number of regions is large, in which case the tour improvement step consumes the most time.

D. DISCUSSIONS

Through both theoretical analyses and simulation studies, we have demonstrated the performance and capability of the proposed TSP-CPP approaches from various aspects. In particular, the DP-based exact approach can guarantee a (near) optimal solution to TSP-CPP, but is not scalable with the number of regions. It is thus suggested to use this approach for small-scale TSP-CPP problems, e.g., less than 17 regions [69]. Moreover, the Fast NN-2Opt generates high-quality tours and is highly efficient even when the number of regions is large. It is thus suitable for cases when the use of DP-based approach is time-prohibitive. In cases when the quality of the solution is of most concern and a relatively long computation time is acceptable, the NN-2Opt, GA-2Opt or Extended GA may be used.

VIII. CONCLUSION

In this paper, we conducted a systematic investigation on a new UAV path planning problem, namely TSP-CPP, which aims to find the optimal tour for a single UAV to fully cover multiple separated regions. To solve this problem, we first investigated the coverage path planning (CPP) for a single convex polygonal region, and developed a CPP method to generate a set of back-and-forth paths, which likely lead to the optimal tour. Based on this CPP method, we then developed a dynamic programming (DP)-based exact approach to solve the TSP-CPP problem, which was proved to be able to find the (near) optimal solution. As the DP-based approach is not scalable with the number of regions, we further developed a heuristic approach, which adopts a two-step iterative procedure to find high-quality tours very efficiently. To illustrate the performance of proposed approaches, we conducted both theoretical analyses and simulation studies comprehensively. The results demonstrated the optimality and efficiency of the proposed approaches in various scenarios. In the future, we will extend the proposed approaches to address more complicated scenarios, such as when static or dynamic obstacles are present. We will also generalize the results to multiple UAVs and explore practical applications of the proposed approaches.

REFERENCES

[1] H. Cruz, M. Eckert, J. Meneses, and J.-F. Martínez, "Efficient forest fire detection index for application in unmanned aerial systems (UASs)," *Sensors*, vol. 16, no. 6, p. 893, 2016.

[2] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1498–1511, Dec. 2016.

[3] L. Ma, L. Cheng, W. Han, L. Zhong, and M. Li, "Cultivated land information extraction from high-resolution unmanned aerial vehicle imagery data," *J. Appl. Remote Sens.*, vol. 8, no. 1, Feb. 2014, Art. no. 083673.
[4] P. Boccardo, F. Chiabrando, F. Dutto, F. Tonolo, and A. Lingua, "UAV deployment exercise for mapping purposes: Evaluation of emergency response applications," *Sensors*, vol. 15, no. 7, pp. 15717–15737, 2015.
[5] J. Xie, L. R. G. Carrillo, and L. Jin, "An integrated traveling salesman and coverage path planning problem for unmanned aircraft systems," *IEEE Control Syst. Lett.*, vol. 3, no. 1, pp. 67–72, Jan. 2019.
[6] J. Xie, L. Jin, and L. R. Garcia Carrillo, "Optimal path planning for unmanned aerial systems to cover multiple regions," in *Proc. AIAA Scitech Forum*, San Diego, CA, USA, Jan. 2019, p. 1794.
[7] I. A. Musliman, A. A. Rahman, and V. Coors, "Implementing 3D network analysis in 3D-GIS," *Int. Arch.*, vol. 37, pp. 1–6, Jul. 2008.
[8] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, and B. Huhnke, "Junior: The stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.
[9] A. Stentz, "Optimal and efficient path planning for partially known environments," in *Intelligent Unmanned Ground Vehicles*, M. H. Hebert, C. Thorpe, and A. Stentz, Eds. Boston, MA, USA: Springer, 1997, pp. 203–220.
[10] A. Nash, S. Koenig, and M. Likhachev, "Incremental phi: Incremental any-angle path planning on grids," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, Pasadena, CA, USA, Jul. 2009, pp. 1824–1830.
[11] Y. Lin and S. Saripalli, "Sampling-based path planning for UAV collision avoidance," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 3179–3192, Nov. 2017.
[12] A. Sánchez and R. Zapata, "Motion planning for car-like robots using lazy probabilistic roadmap method," in *Proc. Mexican Int. Conf. Artif. Intell.* Yucatan, Mexico: Springer, Apr. 2002, pp. 1–10.
[13] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proc. Int. Conf. Robot. Autom.*, Albuquerque, NM, USA, vol. 3, Apr. 1997, pp. 2719–2726.
[14] O. K. Sahingoz, "Flyable path planning for a multi-UAV system with genetic algorithms and bezier curves," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Atlanta, GA, USA, May 2013, pp. 41–48.
[15] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Trans Ind. Informat.*, vol. 9, no. 1, pp. 132–141, Feb. 2013.
[16] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA, USA: Springer, 2017.
[17] J. Kennedy and R. Eberhart, "Particle swarm optimization (PSO)," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, Nov. 1942, pp. 1942–1948.
[18] D. J. Kwak, B. Choi, D. Cho, H. J. Kim, and C.-W. Lee, "Decentralized trajectory optimization using virtual motion camouflage and particle swarm optimization," *Auto. Robots*, vol. 38, no. 2, pp. 161–177, Feb. 2015.
[19] K. Culligan, M. Valenti, Y. Kuwata, and J. P. How, "Three-dimensional flight experiments using on-line mixed-integer linear programming trajectory optimization," in *Proc. Amer. Control Conf.*, New York City, NY, USA, Jul. 2007, pp. 5322–5327.
[20] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using RRT based approaches: A survey and future directions," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 11, pp. 97–107, 2019.
[21] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning: A 2013 survey," in *Proc. Ind. Eng. Syst. Manage. (IESM)*, Rabat, Morocco, Oct. 2013, pp. 1–8.
[22] K. Bollino and L. R. Lewis, "Collision-free multi-UAV optimal path planning and cooperative control for tactical applications," in *Proc. AIAA Guid., Navigat. Control Conf. Exhib.*, Honolulu, HI, USA, Aug. 2008, p. 7134.
[23] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
[24] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ, USA: Princeton Univ. Press, 2006.
[25] J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem," *Oper. Res.*, vol. 11, no. 6, pp. 972–989, 1963.
[26] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," *SIAM Rev.*, vol. 33, no. 1, pp. 60–100, Mar. 1991.
[27] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *J. Oper. Res. Soc. Amer.*, vol. 2, no. 4, pp. 393–410, Nov. 1954.
[28] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *J. ACM*, vol. 9, no. 1, pp. 61–63, Jan. 1962.
[29] C. Chauhan, R. Gupta, and K. Pathak, "Survey of methods of solving TSP along with its implementation using dynamic programming approach," *Int. J. Comput. Appl.*, vol. 52, no. 4, pp. 12–19, 2012.
[30] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," *Local Search Combinat. Optim.*, vol. 1, no. 1, pp. 215–310, 1997.
[31] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht, "Genetic algorithms for the traveling salesman problem," in *Proc. Int. Conf. Genetic Algorithms Appl.*, vol. 160, no. 168. Pittsburgh, PA, USA: Lawrence Erlbaum, Jul. 1985, pp. 160–168.
[32] C.-H. Song, K. Lee, and W. Don Lee, "Extended simulated annealing for augmented TSP and multi-salesmen TSP," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 3. Portland, OR, USA, Jul. 2003, pp. 2340–2343.
[33] C. C. Skiścim and B. L. Golden, "Optimization by simulated annealing: A preliminary computational study for the TSP," in *Proc. 15th Conf. Winter Simulation*, Arlington, VA, USA, vol. 2, Dec. 1983, pp. 523–535.
[34] K.-S. Leung, H.-D. Jin, and Z.-B. Xu, "An expanding self-organizing neural network for the traveling salesman problem," *Neurocomputing*, vol. 62, pp. 267–292, Dec. 2004.
[35] R. E. Burkard, V. G. Deineko, R. van Dal, J. A. A. van der Veen, and G. J. Woeginger, "Well-solvable special cases of the traveling salesman problem: A survey," *SIAM Rev.*, vol. 40, no. 3, pp. 496–546, Jan. 1998.
[36] S. Anbuudayasankar, K. Ganesh, and S. Mohapatra, "Survey of methodologies for TSP and VRP," in *Models for Practical Routing Problems Logistics*. Cham, Switzerland: Springer, 2014, pp. 11–42.
[37] T. Bektas, "The multiple traveling salesman problem: An overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, Jun. 2006.
[38] A. Bretin, G. Desaulniers, and L.-M. Rousseau, "The traveling salesman problem with time windows in postal services," *J. Oper. Res. Soc.*, early access, Jan. 2020, doi: 10.1080/01605682.2019.1678403.
[39] K. Ilavarasi and K. S. Joseph, "Variants of travelling salesman problem: A survey," in *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*, Chennai, India, Feb. 2014, pp. 1–7.
[40] C. Papalitsas, T. Andronikos, K. Giannakis, G. Theocharopoulou, and S. Fanarioti, "A QUBO model for the traveling salesman problem with time windows," *Algorithms*, vol. 12, no. 11, p. 224, 2019.
[41] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis, "VRP with time windows," in *The Vehicle Routing Problem*. Philadelphia, PA, USA: SIAM, 2002, pp. 157–193.
[42] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, part I: Route construction and local search algorithms," *Transp. Sci.*, vol. 39, no. 1, pp. 104–118, Feb. 2005.
[43] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Oper. Res.*, vol. 40, no. 2, pp. 342–354, Apr. 1992.
[44] D. Mester and O. Bräysy, "Active-guided evolution strategies for large-scale capacitated vehicle routing problems," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 2964–2975, Oct. 2007.
[45] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, and D. Naddef, "Separating capacity constraints in the CVRP using tabu search," *Eur. J. Oper. Res.*, vol. 106, nos. 2–3, pp. 546–557, Apr. 1998.
[46] R. Baldacci, A. Mingozzi, and R. Roberti, "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints," *Eur. J. Oper. Res.*, vol. 218, no. 1, pp. 1–6, Apr. 2012.
[47] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 1, pp. 70–85, Jan. 2017.
[48] R. H. Mole, "A survey of local delivery vehicle routing methodology," *J. Oper. Res. Soc.*, vol. 30, no. 3, pp. 245–252, Mar. 1979.
[49] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 3, pp. 345–358, Jun. 1992.
[50] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auto. Syst.*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013.

[51] J. Palacin, T. Palleja, I. Valganon, R. Pernia, and J. Roca, "Measuring coverage performances of a floor cleaning mobile robot using a vision system," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 2005, pp. 4236–4241.
[52] J.-C. Latombe, *Robot Motion Planning*, vol. 124. New York, NY, USA: Springer, 2012.
[53] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*, A. Zelinsky, Ed. London, U.K.: Springer, 1998, pp. 203–209.
[54] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *Int. J. Robot. Res.*, vol. 21, no. 4, pp. 331–344, Apr. 2002.
[55] A. Zelinsky, R. A. Jarvis, J. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proc. Int. Conf. Adv. Robot.*, vol. 13, 1993, pp. 533–538.
[56] Y. Gabriely and E. Rimon, "Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, Washington, DC, USA, May 2002, pp. 954–960.
[57] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "BSA: A complete coverage algorithm," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 2005, pp. 2040–2044.
[58] H. Choset, "Coverage for robotics-a survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, nos. 1–4, pp. 113–126, 2001.
[59] J. I. Vasquez-Gomez, J.-C. Herrera-Lozada, and M. Olguin-Carbajal, "Coverage path planning for surveying disjoint areas," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Dallas, TX, USA, Jun. 2018, pp. 899–904.
[60] J. M. Keil, "Polygon decomposition," *Handbook Comput. Geometry*, vol. 2, pp. 491–518, Jan. 2000.
[61] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres, "Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction," *Expert Syst. Appl.*, vol. 55, pp. 441–451, Aug. 2016.
[62] J. I. Vasquez-Gomez, M. Marciano-Melchor, L. Valentin, and J. C. Herrera-Lozada, "Coverage path planning for 2D convex regions," *J. Intell. Robot. Syst.*, vol. 97, no. 1, pp. 81–94, Jan. 2020.
[63] J. Xie, Y. Wan, B. Wang, S. Fu, K. Lu, and J. H. Kim, "A comprehensive 3-dimensional random mobility modeling framework for airborne networks," *IEEE Access*, vol. 6, pp. 22849–22862, 2018.
[64] Y. Lin and S. Saripalli, "Path planning using 3D dubins curve for unmanned aerial vehicles," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Orlando, FL, USA, May 2014, pp. 296–304.
[65] M. Coombes, W.-H. Chen, and C. Liu, "Boustrophedon coverage path planning for UAV aerial surveys in wind," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Miami, FL, USA, Jun. 2017, pp. 1563–1571.
[66] R. Matai, S. P. Singh, and M. L. Mittal, "Traveling salesman problem: An overview of applications, formulations, and solution approaches," in *Traveling Salesman Problem, Theory and Applications*, D. Davendra, Ed. London, U.K.: IntechOpen, 2010.
[67] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell, "Approximation algorithms for lawn mowing and milling," *Comput. Geometry*, vol. 17, nos. 1–2, pp. 25–50, Oct. 2000.
[68] Y. Li, H. Chen, M. Joo Er, and X. Wang, "Coverage path planning for UAVs based on enhanced exact cellular decomposition method," *Mechatronics*, vol. 21, no. 5, pp. 876–885, Aug. 2011.
[69] M. Bellmore and G. L. Nemhauser, "The traveling salesman problem: A survey," *Oper. Res.*, vol. 16, no. 3, pp. 538–558, Jun. 1968.
[70] J. Kirk. *Traveling Salesman Problem—Nearest Neighbor, MATLAB Central File Exchange*. Accessed: Jan. 6, 2020. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/21297-traveling-salesman-problem-nearest-neighbor
[71] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *Eur. J. Oper. Res.*, vol. 126, pp. 106–130, Oct. 2000.
[72] J. Kirk. *Traveling Salesman Problem—Genetic Algorithm, MATLAB Central File Exchange*. Accessed: Jan. 6, 2020. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm

JUNFEI XIE (Member, IEEE) received the B.S. degree in electrical engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2012, and the M.S. degree in electrical engineering and the Ph.D. degree in computer science and engineering from the University of North Texas (UNT), Denton, TX, USA, in 2013 and 2016, respectively. She was an Assistant Professor with the Department of Computing Sciences, Texas A&M University–Corpus Christi (TAMUCC). She is currently an Assistant Professor with the Electrical and Computer Engineering Department, San Diego State University. Her current research interests include large-scale dynamic system design and control, managing and mining spatiotemporal data, unmanned aerial systems, distributed computing, airborne computing, complex information systems, air traffic flow management, and so on.

LUIS RODOLFO GARCIA CARRILLO (Member, IEEE) was born in Gomez Palacio, Durango, Mexico. He received the B.S. degree in electronic engineering and the M.S. degree in electrical engineering from the Institute of Technology of La Laguna, Coahuila, Mexico, in 2003 and 2007, respectively, and the Ph.D. degree in control systems from the University of Technology of Compiegne, France, in 2011. From 2012 to 2013, he was a Postdoctoral Researcher with the Center for Control, Dynamical Systems and Computation, University of California, Santa Barbara (UCSB). During this time, he was also a Researcher with the UCSB Institute for Collaborative Biotechnologies. He is currently an Assistant Professor with the Department of Electrical Engineering, Texas A&M University–Corpus Christi. He is a coauthor of one patent, more than 65 technical publications, and one book. His research interests have focused on control systems, multiagent systems, game theory, intelligent control, and the use of computer vision in feedback control. He is also an Associate Editor of *Mathematical Problems in Engineering*.

LEI JIN received the B.S. degree in applied mathematics from the Huazhong University of Science and Technology, Wuhan, China, in 1998, the M.S. degree in applied mathematics from Zhejiang University, Hangzhou, China, in 2001, and the Ph.D. degree in statistics from Texas A&M University, College Station, TX, USA, in 2007. He joined the Department of Mathematics and Statistics, Texas A&M University–Corpus Christi, in 2013, as an Assistant Professor, where he is currently an Associate Professor. His research interests include time series and complex data analysis, applied statistics and mathematics in engineering, and so on.