Multiregional Coverage Path Planning for Multiple Energy Constrained UAVs

Junfei Xie, *Senior Member, IEEE*, and Jun Chen, *Member, IEEE*

***Abstract*— In recent years, we have witnessed a growing use of unmanned aerial vehicles (UAVs) in a variety of civil, commercial and military applications. Among these applications, many require the UAVs to scan or survey one or more regions, such as land monitoring, disaster assessment, search and rescue. To realize such applications, path planning is a key step. Although the coverage path planning (CPP) problem for a single region has been extensively studied in the literature, CPP for multiple regions has gained much less attention. This multi-regional CPP problem can be considered as a variant of the (multiple) traveling salesman problem (TSP) enhanced with CPP. Previously, we have studied the case of a single UAV. In this paper, we extend our previous studies to further consider multiple UAVs with energy constraints. To solve this new path planning problem, we develop two approaches: 1) a branch-and-bound (BnB) based approach that can find (near) optimal tours and 2) a genetic algorithm (GA) based approach that can solve large-scale problems efficiently under different objectives. Comprehensive theoretical analyses and computational experiments demonstrate the promising performance of the proposed approaches in terms of optimality and efficiency.**

***Index Terms*— Coverage path planning, traveling salesman problem, branch and bound, genetic algorithm, multiple regions, multiple unmanned aerial vehicles.**

I. Introduction

IN the past few years, unmanned aerial vehicles (UAVs) have experienced an unprecedented level of growth. Many of their applications, such as search and rescue [1], land surveying [2], precision agriculture [3], disaster assessment [4], to name a few, involve the task of planning the path for the UAVs to scan a region completely. This task can be formulated as the coverage path planning (CPP) problem [5] that seeks the optimal path with the minimum cost to cover an area.

Despite the abundant studies on CPP for a single region [5], the problem of how to plan the path to cover multiple disjoint regions has not been well studied. This problem arises in many real applications such as emergency response, pasture management and land monitoring, in which the areas to be inspected, assessed or monitored often range over multiple spatially separated regions. This multi-regional CPP problem can be considered as a variant of the traveling salesman problem (TSP) [6], [7] enhanced with CPP, where the determination of the region visiting order is a TSP and the coverage for each region is a CPP. We hence name it TSP-CPP. Despite the existence of many approaches for TSP and CPP, TSP-CPP cannot be solved by a direct extension to any of these TSP or CPP approaches. This is because solving TSP-CPP requires the determination of the entrance and exit locations in each region, which impact both the region visiting order and intra-regional coverage paths and are not considered in either TSP or CPP. To solve TSP-CPP, the region visiting order, the path to cover each region, and the entrance and exit locations in each region should be *jointly* optimized.

A related problem, called the tour polygon problem (TPP) [8], [9] also considers multiple regions, but it does not require each region to be covered. In particular, the TPP seeks the optimal path to merely visit multiple regions. In cases when the UAV is only allowed to visit the edge of each region without entering the region, the TPP is also known as the zookeeper problem [10]. Otherwise, if the UAV can freely cross the regions, the TPP is often referred to as the Safari problem [11].

In this paper, we consider a more complicated problem where multiple UAVs cooperate to fully cover multiple spatially distributed regions. Moreover, as UAVs have limited power supplies and thus bounded flight ranges, we also take this realistic energy constraint into the consideration. The resulting problem is named as the Energy constrained Multiple TSP-CPP (EMTSP-CPP). To the best of our knowledge, this problem hasn't been systematically investigated in the literature.

The idea of using multiple agents to share the workload so as to reduce the task completion time is not new. The multi-robot CPP and the multiple TSP (MTSP) are a direct extension of the CPP and TSP to multiple agents, respectively, both of which have been extensively studied [5], [12], [13]. The vehicle routing problem (VRP) is a generalization of MTSP that also considers various constraints to solve real-life delivery problems [14]–[21]. Depending on the constraints considered, VRP has many variants, such as the VRP with time window (VRPTW), capacitated VRP (CVRP), multiple depots VRP (MDVRP), and distance-constrained VRP (DVRP), to name a few. Among these variants, DVRP is the most relevant to this study, which aims to find the optimal tours for a set of vehicles to visit a set of target locations,

Manuscript received 3 October 2020; revised 11 April 2021; accepted 14 March 2022. Date of publication 28 March 2022; date of current version 11 October 2022. This work was supported in part by the National Science Foundation under Grant CI-1953048 and CAREER-2048266 and in part by San Diego State University under the University Grants Program. The Associate Editor for this article was S. Rathinam. *(Corresponding author: Junfei Xie.)*

Junfei Xie is with the Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA 92182 USA (e-mail: jxie4@sdsu.edu).

Jun Chen is with the Department of Aerospace Engineering, San Diego State University, San Diego, CA 92182 USA (e-mail: jun.chen@sdsu.edu).

Digital Object Identifier 10.1109/TITS.2022.3160402

with the length of each tour not exceeding a maximum tour length. This problem can be considered as a special case of the EMTSP-CPP if all regions are small enough so that they can be fully covered by merely visiting a single location within the region. Nevertheless, the general EMTSP-CPP is much more challenging than DVRP and other variants of VRP, due to the involvement of CPP. Particularly, in VRP and its variants, the distance between any two target locations is known and fixed. However, in EMTSP-CPP, the distance from one region to another is unknown and varies with the change of the region visiting order and intra-regional coverage paths. Such interactions among inter-regional distances, region visiting order and intra-regional coverage paths significantly complicate the problem.

In this paper, we conduct a systematic investigation on EMTSP-CPP, and develop two approaches to solve this new problem. The main contributions are summarized as follows:

- *A Novel Branch-and-Bound Method that Finds (Near) Optimal Solutions to Small-Scale EMTSP-CPP.* The proposed branch-and-bound (BnB) method builds a binary tree to search for the best region visiting order that leads to the shortest tour, where tours are constructed progressively using a heuristic. This approach is proved to be able to find (near) optimal tours under certain assumptions.
- *A New Genetic Algorithm that Solves Large-scale EMTSP-CPP Efficiently.* The proposed GA is featured by a new sets-based chromosome design that speeds up the convergence, as well as new crossover and mutation operators, and a constraint-aware fitness function. By adjusting the fitness function, the proposed GA can be used to achieve different objectives, such as minimizing the total cost and balancing the workload among UAVs.
- *Comprehensive Theoretical Studies and Computational Experiments.* Through theoretical studies, we prove the validity of the lower-bound calculation in the proposed BnB method and also prove the optimality of the derived solution. We also conduct comprehensive computational experiments to evaluate the optimality and efficiency of the proposed two approaches. The results demonstrate their promising performances.

The rest of the paper is organized as follows. Section II reviews existing work relevant to our study. In Section III, we provide a possible mathematical formulation for EMTSP-CPP. Section IV briefly reviews a CPP method and an efficient heuristic approach for TSP-CPP. Based on these methods, we then solve EMTSP-CPP. In particular, the BnB method is described in Section V, along with the theoretical results on its performance. The GA is introduced in Section VI. In Section VII, we conduct computational experiments to evaluate the performance of the proposed two approaches. Section VIII concludes the paper with a brief discussion on future works.

II. Related Work

In this section, we review existing works relevant to EMTSP-CPP considered in this study.

EMTSP-CPP has been rarely studied. A simpler problem without energy constraint is considered in [22], which introduces a heuristic procedure that first assigns a subset of regions to each UAV and then optimizes the region visiting order. This study, however, oversimplifies the coverage problem. How to enter or exit each region and what is the path to fully cover each region are not addressed. Another related work is presented in [23], which considers the distributed motion planning problem for multiple robots to cover multiple rectangular regions. In this study, each robot determines its next motion based on a set of rules.

TSP-CPP is a special case of EMTSP-CPP with a single UAV. In our previous studies [24]–[26], we developed multiple approaches for TSP-CPP including a dynamic programming (DP) based approach, a grid-based approach and a nearest neighbor (NN) and 2-Opt based heuristic approach, called Fast NN-2Opt. The DP- and grid-based approaches are proved to be able to find optimal solutions under some mild assumptions and the Fast NN-2Opt achieves a good tradeoff between optimality and efficiency. In [27], a two steps path planning (TSPP) approach was introduced, which first determines the region visiting order using regions' centroids and then plans the path to cover each region. Although simple, this heuristic approach produces longer tours, compared with our methods [26], as it ignores the interaction between the region visiting order and intra-regional coverage paths. It is also less efficient than the Fast NN-2Opt algorithm.

In the special case when all regions are sufficiently small so that they can be fully covered by UAV's camera footprint at their centroids, the EMTSP-CPP is reduced to a DVRP. The most frequently used exact approach for DVRP is the branch-and-bound method [28], [29]. An integer linear programming algorithm is introduced in [30] to solve VRP with both distance and capacity constraints. In [31], the DVRP is transformed into a MTSP with time windows and solved using a column generation method. Different integer linear programming formulations of DVRP are presented in [32]–[35]. Other exact approaches for DVRP include the cutting planes algorithm [36] and the lexisearch algorithm [37]. There are also some approximate and heuristic approaches for DVRP, such as the 2-approximation algorithm [38], constant-factor differential approximation algorithm [39] and variable neighborhood search [40]. Also of relevance, a generalization of DVRP that further considers multiple fuel stations to allow a vehicle to refuel was investigated in [41]–[43].

Multi-robot CPP is a special case of EMTSP-CPP with a single region. Most existing approaches for this problem are extended from the methods for single-robot CPP by using a strategy to divide the workload. For instance, in [44], a greedy auction heuristic is used to allocate workload among the robots and the Boustrophedon single-robot coverage algorithm is used to plan the paths for each robot. In [45], a task scheduler is first used to partition the target area into non-overlapping sub-areas and the wavefront algorithm is then applied to cover each sub-area. In [46], the spanning tree coverage algorithm for a single robot is extended to multiple robots. There are also approaches that directly solve the multi-robot CPP without relying on any single-robot algorithms, such as the bio-inspired coverage algorithms introduced in [47], [48].

Other problems that are related to EMTSP-CPP include CPP, TSP, MTSP, VRP and their variants. For a complete review of existing algorithms for these problems, interested readers are referred to the survey papers [5], [7], [12]–[17], [49]–[53].

III. Problem Formulation

In this section, we first describe the EMTSP-CPP to be solved. For the sake of clarity, we then provide a mathematical formulation for this problem.

A. Problem Description and Assumptions

Consider the scenario where $M \in \mathbb{Z}^+$ multirotor UAVs of the same type are assigned to scan or assess $N \in \mathbb{Z}^+$ convex polygonal regions distributed over the space without overlap. The location, size and shape of each region i, $i \in [N]$, are known, which are captured by its vertices $P_i \in \mathbb{R}^{v_i \times 2}$, where the jth row of P_i stores the location of the jth vertex of region i, $v_i \in \mathbb{Z}^+$ is region i's total number of vertices, and $[n]$ denotes the set $\{1, 2, \ldots, n\}$ for any $n \in \mathbb{Z}^+$. All UAVs depart from the same depot and return to this depot after the mission is completed. The location of the depot is known and denoted as $p_0 \in \mathbb{R}^{1\times 2}$, where number 0 is the label of the depot.

Each UAV carries a sensor (e.g., camera) with a sensing range of $l \times w$, where $l, w > 0$ are known constants. For simplicity, we assume that each UAV flies at a constant altitude, so that the path planning problem can be formulated in the 2-dimensional (2-D) space. We also assume each UAV flies at a constant speed V and can turn with an arbitrary radius of curvature at speed V. In addition, we assume the maximum distance each UAV can travel at speed V, due to limited power supply, is D_{max}. We also assume each region is scanned by a single UAV and D_{max} is sufficiently large for covering each region individually. The EMTSP-CPP is then concerned with finding the optimal tours[1] for the M UAVs to fully cover all regions, while satisfying the energy constraints of the UAVs.

B. Decision Variables and Constraints

To describe the visiting order for the regions, we introduce a binary decision variable x_{ij}, $i, j \in [N] \cup \{0\}$, which equals 1 if a UAV visits region (or depot) j after region (or depot) i, and equals 0 otherwise. The following constraints then ensure each UAV departs from and returns to the same depot and each region is scanned only once.

$$\sum_{i\in[N]} x_{0i} = M \tag{1}$$

$$\sum_{i\in[N]} x_{i0} = M \tag{2}$$

$$\sum_{i\in[N]} x_{ij} = 1, \quad \forall j \in [N] \tag{3}$$

[1]A tour is a feasible and complete path for an UAV to cover all regions assigned to it.

$$\sum_{j\in[N]} x_{ij} = 1, \quad \forall i \in [N] \tag{4}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \le |S| - 1, \quad \forall S \subseteq [N], \quad S \neq \emptyset \tag{5}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in [N] \cup \{0\} \tag{6}$$

In particular, constraints (1)-(2) ensure that exactly M UAVs depart from the depot as mission starts and return to the same depot after the mission is completed. Constraints (3)-(4) ensure that each region is scanned once. Constraint (5) prevents sub-tours, where $|S|$ represents the cardinality of set S. Constraint (6) specifies the possible values x_{ij} can take.

Furthermore, to describe the coverage path for each region, we introduce variable $\boldsymbol{b}_i = (b_{ik})_{k=1}^{n_i} \in \boldsymbol{B}_i$ to denote a coverage path for region $i \in [N]$ such that, by following this path, the UAV will cover the region completely. Here b_{ik} is the kth waypoint in the path, $n_i \in \mathbb{Z}^+$ is the total number of waypoints, and $\boldsymbol{B}_i$ is the full set of possible coverage paths for region i. b_{i1} and b_{in_i} then capture the entrance and exit locations at region i, respectively. A feasible set of M tours that fully cover all regions can then be generated by connecting the coverage paths $\boldsymbol{b}_i$, $\forall i \in [N]$, and the depot p_0, according to the order specified by x_{ij}, $\forall i, j \in [N] \cup \{0\}$. Particularly, denote $\boldsymbol{\tau} \in \boldsymbol{T}$ as a tour, where $\boldsymbol{T}$ is the full set of tours and $|\boldsymbol{T}| = M$. The set of regions covered by each tour $\boldsymbol{\tau}$, denoted as $S_{\boldsymbol{\tau}}$, satisfies the following condition

$$\sum_{i\in S_{\boldsymbol{\tau}}}\sum_{j\in S_{\boldsymbol{\tau}}} x_{ij} = |S_{\boldsymbol{\tau}}| - 1, S_{\boldsymbol{\tau}} \subseteq [N], |S_{\boldsymbol{\tau}}| > 1$$

x_{ij}, $\forall i, j \in S_{\boldsymbol{\tau}}$, then capture the visiting order for regions $S_{\boldsymbol{\tau}}$.

As the maximum distance each UAV can travel is D_{max}, the length of each tour $\boldsymbol{\tau}$ should not exceed this upper bound. The following constraint should thus be satisfied:

$$D_{\boldsymbol{\tau}} \le D_{max}, \quad \forall \boldsymbol{\tau} \in \boldsymbol{T} \tag{7}$$

where $D_{\boldsymbol{\tau}}$ is the length of the tour $\boldsymbol{\tau}$. It can be computed by:

$$D_{\boldsymbol{\tau}} = \sum_{i\in S_{\boldsymbol{\tau}}} \left[x_{0i}\,\mathrm{d}(p_0, b_{i1}) + x_{i0}\,\mathrm{d}(b_{in_i}, p_0) \right] + \sum_{i\in S_{\boldsymbol{\tau}}}\sum_{j\in S_{\boldsymbol{\tau}}} x_{ij}\,\mathrm{d}(b_{in_i}, b_{j1}) + \sum_{i\in S_{\boldsymbol{\tau}}} \mathrm{g}(\boldsymbol{b}_i)$$

where $\mathrm{d}(a, b)$ is the Euclidean distance to travel from location a to location b, and $\mathrm{g}(\boldsymbol{b}_i) = \sum_{k=1}^{n_i-1} \mathrm{d}(b_{ik}, b_{i(k+1)})$ computes the length of the coverage path $\boldsymbol{b}_i$.

C. Mathematical Formulation

In this study, we consider the EMTSP-CPP with different objectives. The first objective is to minimize the total travel distance, given the number of UAVs M, which can be formulated as:

$$\mathcal{P}_1 : \underset{\substack{x_{ij}, \forall i,j \in [N]\cup\{0\} \\ \boldsymbol{b}_i \in \boldsymbol{B}_i, \forall i \in [N]}}{\text{minimize}} \; J_1 = \sum_{\boldsymbol{\tau}\in\boldsymbol{T}} D_{\boldsymbol{\tau}} \quad \text{subject to: Constraints (1)-(7)} \tag{8}$$

The second objective is to balance the workload among the UAVs, given the number of UAVs M, which can be achieved

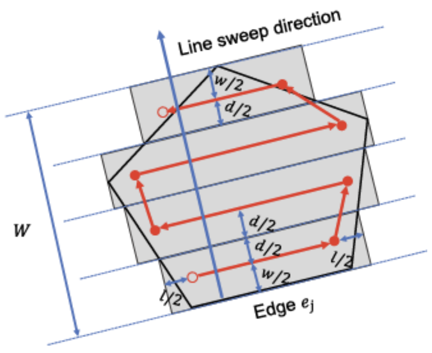


Fig. 1. A BFP (red line) that covers a pentagonal region.

by minimizing the length of the longest tour. The mathematical formulation of this problem is given by:

$$\mathcal{P}_2: \underset{\substack{x_{ij},\forall i,j\in[N]\cup\{0\}\\ \boldsymbol{b}_i\in\boldsymbol{B}_i,\forall i\in[N]}}{\text{minimize}} \quad J_2=\max_{\tau\in T} D_\tau$$
$$\text{subject to: Constraints (1)-(7)} \qquad (9)$$

Note that both $\mathcal{P}_1$ and $\mathcal{P}_2$ cannot be directly solved, e.g., using mixed-integer linear programming (MILP) solvers, as the coverage paths $\boldsymbol{b}_i$ are unknown variables that need to be determined and there are infinite number of possible coverage paths for each region, i.e., $\boldsymbol{B}_i$ is an infinite set. In this study, we introduce two approximate approaches to solve the EMTSP-CPP with different scales. Both approaches sample the search space and limit the search for optimal $\boldsymbol{b}_i$ to a finite set, denoted as $\hat{\boldsymbol{B}}_i$. In the following sections, we first introduce a back-and-forth coverage path planning (BF-CPP) algorithm developed in our previous studies [26] to determine $\hat{\boldsymbol{B}}_i$. Based on this algorithm, we then solve the EMTSP-CPP.

IV. Preliminaries

In this section, we first briefly review the BF-CPP algorithm [26]. We then describe the Fast NN-2Opt algorithm [26], which was developed in our previous study for TSP-CPP and will be used in the proposed GA to solve EMTSP-CPP.

A. BF-CPP

To determine $\hat{\boldsymbol{B}}_i$ for each region $i \in [N]$, we apply the BF-CPP algorithm [26], which generates a set of paths with back-and-forth patterns. Each path has a *line sweep direction* perpendicular to one of the region's edges (see Fig. 1). The procedure of BF-CPP is briefly summarized in Algorithm 1. In particular, for each edge of a region, the region is first decomposed into a set of sub-regions (Lines 2-7). The shortest line segment that fully covers each sub-region is then found for each sub-region (Lines 8-9). Finally, the line segments in adjacent sub-regions are connected to generate the back-and-forth paths (BFPs). These BFPs have nice properties in terms of full coverage guarantee, optimality, and complexity. For more details about BF-CPP, please refer to [26].

Note that if we pre-store $\hat{\boldsymbol{B}}_i$ generated by BF-CPP for each region $i \in [N]$, we may then formulate the search for the

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

```
1  foreach edge $e_j$ of region $i$ do
2      $W \leftarrow$ span of edge $e_j$;
3      if $\lceil W/w \rceil > 1$ then
4          $d \leftarrow \frac{W-w}{\lceil W/w \rceil - 1}$;
5      else
6          $d \leftarrow 0$;
7      Apply cellular decomposition to decompose region $i$
       into a set of sub-regions along the line sweep
       direction perpendicular to edge $e_j$, where the width of
       the first and last sub-regions is $w$ and the width of
       intermediate sub-regions is $\frac{w+d}{2}$;
8      foreach sub-region $k$ do
9          Find the shortest line segment that leads to full
           coverage of sub-region $k$ and is $d$ distance away
           from the segments in adjacent sub-regions if any;
10     Connect line segments in adjacent sub-regions to
       generate all possible BFPs;
11     $\hat{\boldsymbol{B}}_i \leftarrow \{\hat{\boldsymbol{B}}_i$, BFPs$\}$;
12 return $\hat{\boldsymbol{B}}_i$.
```

optimal tours as a MILP problem, e.g., by introducing a binary decision variable for each coverage path $\boldsymbol{b}_i \in \hat{\boldsymbol{B}}_i$, $i \in [N]$, that takes value 1 when $\boldsymbol{b}_i$ is selected and equals 0 otherwise, as well as adding constraints to limit the number of coverage paths for each region to one. However, the resulting problem is too complex to be solved using solvers like CPLEX [54] and Gurobi [55] within a reasonable time, considering the large number of decision variables, which is $N^2 + \sum_{i=1}^{N} |\hat{\boldsymbol{B}}_i|$.[2] Furthermore, the distance matrix that stores the distances between the end points of each pair of coverage paths from different regions can be huge, whose size is $O(N^2 \prod_{i=1}^{N} |\hat{\boldsymbol{B}}_i|)$. Therefore, in this study, we explore other approaches to solve EMTSP-CPP efficiently.

B. Fast NN-2Opt

The Fast NN-2Opt [26] is a simple and efficient heuristic approach for TSP-CPP, which finds high-quality solutions by performing two phases as summarized in Algorithm 2. In the first phase, the tour is initialized using a nearest neighbor (NN) based algorithm. The key idea is to first determine the region visiting order using the NN algorithm based on the regions' centroids (Line 4), and then find the complete tour that fully covers all regions using function FindTour() in Line 5, where the pseudocode for function FindTour() is provided in Algorithm 3. In this function, the tour is generated by connecting the best BFP for each region, which minimizes the length of the path constructed so far, one by one in the region visiting order (Line 8).

In the second phase, the tour is improved iteratively until convergence by using a 2-Opt based algorithm (function ImproveTour() in Line 6). The pseudocode for function

[2]The number of coverage paths $|\hat{\boldsymbol{B}}_i|$ generated by BF-CPP is up to $4v_i$ (see ref. [6]).

Algorithm 2 Fast NN-2Opt Algorithm for TSP-CPP

```
  Input: $p_0$, $\{P_i\}$, $l$, $w$
  Output: tour $\boldsymbol{\tau}^*$, cost $J^*$
1 for $i \leftarrow 1$ to $N$ do
2     $\hat{\boldsymbol{B}}_i \leftarrow$BF-CPP($P_i$, $l$, $w$);
3     $c_i \leftarrow$ centroid of region $i$;
  /* Initialize the tour */
4 Apply NN algorithm to find the region visiting order $\boldsymbol{o}$
  using regions' centroids;
5 $\boldsymbol{\tau} \leftarrow$ FINDTOUR($\boldsymbol{o}, p_0, \{c_i\}, \{\hat{\boldsymbol{B}}_i\}$);
  /* Improve the tour */
6 $\boldsymbol{\tau}^* \leftarrow$ IMPROVETOUR($\boldsymbol{\tau}, \boldsymbol{o}, p_0, \{c_i\}, \{\hat{\boldsymbol{B}}_i\}$);
7 $J^* \leftarrow \mathrm{g}(\boldsymbol{\tau}^*)$;
8 return $\boldsymbol{\tau}^*$, $J^*$
```

Algorithm 3 FINDTOUR($\boldsymbol{o}$, p_0, $\{c_i\}$, $\{\hat{\boldsymbol{B}}_i\}$)

```
1 $\boldsymbol{\tau} \leftarrow \{p_0\}$;
2 for $t \leftarrow 1$ to $N$ do
3     $i \leftarrow \boldsymbol{o}(t)$;
4     if $t < N$ then
5         $j \leftarrow \boldsymbol{o}(t+1)$;
6     else
7         $j \leftarrow 0$;
8     $\boldsymbol{b}_i \leftarrow$ coverage path for region $i$ that minimizes
      $\mathrm{g}(\boldsymbol{b}_i) + \mathrm{d}(\boldsymbol{\tau}(\text{end}), \boldsymbol{b}_{i1}) + \mathrm{d}(\boldsymbol{b}_{in_i}, c_j)$, where $\boldsymbol{\tau}(\text{end})$ is
      the last location in path $\boldsymbol{\tau}$, $c_0 = p_0$ and $\boldsymbol{b}_i \in \hat{\boldsymbol{B}}_i$;
9     $\boldsymbol{\tau} \leftarrow \{\boldsymbol{\tau}, \boldsymbol{b}_i\}$;
10 $\boldsymbol{\tau} \leftarrow \{\boldsymbol{\tau}, p_0\}$;
11 return $\boldsymbol{\tau}$
```

Algorithm 4 IMPROVETOUR($\boldsymbol{\tau}$, $\boldsymbol{o}$, p_0, $\{c_i\}$, $\{\hat{\boldsymbol{B}}_i\}$)

```
1 $cost \leftarrow \infty$;
2 while $g(\boldsymbol{\tau}) < cost$ do
3     $cost \leftarrow \mathrm{g}(\boldsymbol{\tau})$;
4     for $i \leftarrow 0$ to $N-1$ do
5         for $j \leftarrow i+2$ to $N+1$ do
6             if $d(c_{\boldsymbol{o}(i)}, c_{\boldsymbol{o}(i+1)}) > d(c_{\boldsymbol{o}(i)}, c_{\boldsymbol{o}(j)})$ then
7                 $\boldsymbol{o}' \leftarrow \boldsymbol{o}$ 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;
8                 $\boldsymbol{\tau}' \leftarrow$ FINDTOUR($\boldsymbol{o}', v_0, \{c_i\}, \{\boldsymbol{B}_i\}$);
9                 if $g(\boldsymbol{\tau}') < g(\boldsymbol{\tau})$ then
10                    $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau}'$;
11 return $\boldsymbol{\tau}$
```

IMPROVETOUR() is provided in Algorithm 4. In this function, a 2-Opt move is performed in each iteration to update the region visiting order, if the cost of the resulting tour is reduced.

V. Branch-and-Bound Method for Solving $\mathcal{P}_1$

In this section, we introduce a branch-and-bound (BnB) method for solving EMTSP-CPP with the objective of minimizing the total distance (i.e., $\mathcal{P}_1$).

BnB [56] is a popular exact method for solving integer programming problems and has been widely used to solve TSP, MTSP and VRP. It searches for the optimal solution by constructing a search tree, with the root node representing the original problem and all other nodes representing the subproblems. Each node in the tree is tagged with a lower bound (LB) of the solution value to the associated problem, which is usually obtained by relaxing some constraints. The tree also maintains an upper bound (UB) of the solution value to the original problem, which is updated when a better feasible solution is found. The algorithm selects a node for branching at each iteration and stops iterating when no better feasible solutions can be found. The key ingredients of BnB thus include [57]:

- The *branching rule* that specifies how to partition the current problem into subproblems.
- The *lower bounding strategy* that finds the lower bound of the solution value to any (sub)problem.
- The *upper bounding strategy* that describes how to find a feasible solution to the original problem.
- The *search strategy* that determines which subproblem should be expanded next.

In the following subsections, we describe how each of the ingredients of BnB is designed to solve $\mathcal{P}_1$.

A. Branching Rule

In EMTSP-CPP, the distance between any two regions is an unknown variable that is impacted by multiple factors including the region visiting order and the entrance and exit locations in the two regions. Because of this, the existing BnB methods for TSP, MTSP or VRP [28], [29], [36], [56], [58], which are developed based on the distance matrix, cannot be directly applied to solve EMTSP-CPP. To address this challenge, we first adopt the procedure introduced in [29], [59] to transform the EMTSP-CPP into a TSP-CPP by adding $M-1$ "virtual" copies of the depot located at p_0, but with depot-to-depot distances set to infinity for preventing such travels. These additional $M-1$ "virtual" depots are treated as "regions" to be visited and labeled as $N+1, N+2, \ldots, N+M-1$. Solving any EMTSP-CPP is then equivalent to solving a TSP-CPP on the transformed graph with energy constraints applied on paths that start and end at the depots. After transformation, we then construct a binary search tree to find the best region visiting order and use the idea of function FINDTOUR() to construct the tour progressively. The energy constraints are checked during the search to ensure feasibility of the derived solution.

The binary search tree partitions the original problem into many subproblems by including or excluding certain region-to-region links. As illustrated in Fig. 2, the original problem (node 1) has a solution space that includes all tours, and is first partitioned into two subproblems. Particularly, the solution

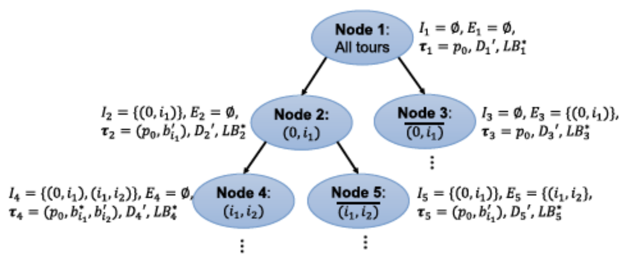

Node 1: All tours
$I_1 = \emptyset, E_1 = \emptyset,$
$\boldsymbol{\tau}_1 = p_0, D_1', LB_1^*$
Node 2: $(0, i_1)$
$I_2 = \{(0,i_1)\}, E_2 = \emptyset,$
$\boldsymbol{\tau}_2 = (p_0, b_{i_1}'), D_2', LB_2^*$
Node 3: $\overline{(0, i_1)}$
$I_3 = \emptyset, E_3 = \{(0,i_1)\},$
$\boldsymbol{\tau}_3 = p_0, D_3', LB_3^*$
Node 4: (i_1, i_2)
$I_4 = \{(0,i_1),(i_1,i_2)\}, E_4 = \emptyset,$
$\boldsymbol{\tau}_4 = (p_0, b_{i_1}^*, b_{i_2}'), D_4', LB_4^*$
Node 5: $\overline{(i_1, i_2)}$
$I_5 = \{(0,i_1)\}, E_5 = \{(i_1,i_2\},$
$\boldsymbol{\tau}_5 = (p_0, b_{i_1}'), D_5', LB_5^*$

Fig. 2. Illustration of the BnB binary search tree.

space of subproblem 2 (node 2 with $(0, i_1)$) includes all tours that contain region-to-region link $(0, i_1)$, and the solution space of subproblem 3 (node 3 with $\overline{(0, i_1)}$) includes all tours that do not contain link $(0, i_1)$, where $i_1, i_2 \in [N+M-1]$ are the labels of the regions. Then, in this example, subproblem 2 is further partitioned into two subproblems. Particularly, the solution space of subproblem 4 (node 4 with (i_1, i_2)) includes all tours that contain both links $(0, i_1)$ and (i_1, i_2), and the solution space of subproblem 5 (node 5 with $\overline{(i_1, i_2)}$) includes all tours that contain link $(0, i_1)$ but not (i_1, i_2). When this partition (or branching) process is carried far enough, some subproblem will eventually have a single solution. In Fig. 2, the information stored in each node is also shown next to the node, which will be explained shortly.

To denote the set of region-to-region links included into and excluded from the solution to subproblem k, we use symbol I_k and E_k, respectively. Then subproblem k can be defined by conditions:

$$x_{ij} = \begin{cases} 1, & (i,j) \in I_k \\ 0, & (i,j) \in E_k \end{cases} \tag{10}$$

The set I_k is constructed in a way such that it also indicates the visiting order for the regions included so far starting from the depot, i.e., $I_k = \{(0, i_1), (i_1, i_2), (i_2, i_3), \ldots, (i_{h-1}, i_h)\}$, where $i_r \in [N+M-1]$, $\forall r \in [h]$, $i_r \neq i_s$, $\forall r \neq s$, and $h = |I_k|$. The links in set I_k are ordered, with the newly inserted link placed at the end.

At each iteration, a subproblem is selected for expansion and two successors are generated. Specifically, if subproblem k is selected for expansion, and node $k+1$ and $k+2$ are its two successors, then,

$$\begin{aligned} I_{k+1} &= I_k \cup \{(i_h, i_{h+1})\} \\ E_{k+1} &= E_k \end{aligned} \tag{11}$$

and

$$\begin{aligned} I_{k+2} &= I_k \\ E_{k+2} &= E_k \cup \{(i_h, i_{h+1})\} \end{aligned} \tag{12}$$

where $i_{h+1} \in [N+M-1] \setminus \{i_1, i_2, \ldots, i_h\}$ is the next region to be examined. Inspired by the branching rule described in [56], we choose i_{h+1} such that excluding link (i_h, i_{h+1}) will cause the lower bound of subproblem k to increase the most. We leave the discussion on how to compute the lower bound to the next subsection. If multiple regions are found, we select one from these regions randomly.

As set I_k provides information about the region visiting order, it can be utilized to construct a partial tour, denoted as $\boldsymbol{\tau}_k$, that covers all regions appeared in I_k, by following a similar procedure shown in Algorithm 3. In particular, starting from the root node, the path stored at this node is just $\boldsymbol{\tau}_1 = p_0$, indicating that all feasible tours start from the depot. The left child of the root node with $I_2 = \{(0, i_1)\}$ then has $\boldsymbol{\tau}_2 = (p_0, \boldsymbol{b}_{i_1}')$, which is the shortest path to travel from p_0 and fully cover region i_1, where

$$\boldsymbol{b}_{i_1}' = \operatorname*{argmin}_{\boldsymbol{b}_{i_1} \in \hat{\boldsymbol{B}}_{i_1}} \mathrm{d}(p_0, b_{i_1 1}) + \mathrm{g}(\boldsymbol{b}_{i_1}) \tag{13}$$

The right child of the root node with $I_3 = \emptyset$ has $\boldsymbol{\tau}_3 = p_0$.

Next, suppose node 4 is the left child of node 2 and $I_4 = I_2 \cup \{(i_1, i_2)\}$, indicating that all solutions to this subproblem must start from depot p_0 and then cover regions i_1 and i_2, we then construct the path based on I_4 and $\boldsymbol{\tau}_2$ at the parent node. In particular, $\boldsymbol{\tau}_4 = (p_0, \boldsymbol{b}_{i_1}^*, \boldsymbol{b}_{i_2}')$, where

$$\boldsymbol{b}_{i_1}^* = \operatorname*{argmin}_{\boldsymbol{b}_{i_1} \in \hat{\boldsymbol{B}}_{i_1}} \mathrm{d}(p_0, b_{i_1 1}) + \mathrm{g}(\boldsymbol{b}_{i_1}) + \mathrm{d}(b_{i_1 n_{i_1}}, c_{i_2}) \tag{14}$$

$$\boldsymbol{b}_{i_2}' = \operatorname*{argmin}_{\boldsymbol{b}_{i_2} \in \hat{\boldsymbol{B}}_{i_2}} \mathrm{d}(b_{i_1 n_{i_1}}^*, b_{i_2 1}) + \mathrm{g}(\boldsymbol{b}_{i_2}) \tag{15}$$

and c_i denotes the centroid of region i. Note that the coverage path for region i_1 is updated with information about the next region to visit. Since which region to visit after region i_2 is unknown, we only base on its previous region i_1 to approximate the best coverage path for region i_2.

In general, suppose $\boldsymbol{\tau}_k = (p_0, \boldsymbol{b}_{i_1}^*, \ldots, \boldsymbol{b}_{i_{h-1}}^*, \boldsymbol{b}_{i_h}')$ is the path stored at node k and node $k+1$ is its left child with $I_{k+1} = I_k \cup \{(i_h, i_{h+1})\}$, then path $\boldsymbol{\tau}_{k+1} = (p_0, \boldsymbol{b}_{i_1}^*, \ldots, \boldsymbol{b}_{i_{h-1}}^*, \boldsymbol{b}_{i_h}^*, \boldsymbol{b}_{i_{h+1}}')$ is stored at this node, where

$$\boldsymbol{b}_{i_h}^* = \operatorname*{argmin}_{\boldsymbol{b}_{i_h} \in \hat{\boldsymbol{B}}_{i_h}} \mathrm{d}(b_{i_{h-1} n_{i_{h-1}}}^*, b_{i_h 1}) + \mathrm{g}(\boldsymbol{b}_{i_h}) + \mathrm{d}(b_{i_h n_{i_h}}, c_{i_{h+1}}) \tag{16}$$

$$\boldsymbol{b}_{i_{h+1}}' = \operatorname*{argmin}_{\boldsymbol{b}_{i_{h+1}} \in \hat{\boldsymbol{B}}_{i_{h+1}}} \mathrm{d}(b_{i_h n_{i_h}}^*, b_{i_{h+1} 1}) + \mathrm{g}(\boldsymbol{b}_{i_{h+1}}) \tag{17}$$

The right child, node $k+2$ with $I_{k+2} = I_k$, has $\boldsymbol{\tau}_{k+2} = \boldsymbol{\tau}_k$. In the special case when $i > N$, $\boldsymbol{b}_i \in \hat{\boldsymbol{B}}_i = \{p_0\}$, i.e., $n_i = 1$, $b_{i1} = c_i = p_0$ and $\mathrm{g}(\boldsymbol{b}_i) = 0$. In addition, if I_{k+1} includes all regions, i.e., $\{i_1, i_2, \ldots, i_{h+1}\} = [N+M-1]$, which indicates that $(\boldsymbol{\tau}_{k+1}, p_0)$ is a complete tour, we let

$$\begin{aligned} \boldsymbol{b}_{i_{h+1}}' &= \boldsymbol{b}_{i_{h+1}}^* \\ &= \operatorname*{argmin}_{\boldsymbol{b}_{i_{h+1}} \in \hat{\boldsymbol{B}}_{i_{h+1}}} \mathrm{d}(b_{i_h n_{i_h}}^*, b_{i_{h+1} 1}) + \mathrm{g}(\boldsymbol{b}_{i_{h+1}}) \\ &\quad + \mathrm{d}(b_{i_{h+1} n_{i_{h+1}}}, p_0) \end{aligned} \tag{18}$$

B. Bounding Strategies

The objective of bounding is to prune the nodes in the search tree as many as possible. A node can be pruned if its lower bound exceeds the value of the best solution found so far. The tighter the lower bound is, the faster the optimal solution can be found. To find the lower bound for EMTSP-CPP, we construct a matrix $\mathcal{D} \in R^{(N+M)\times(N+M)}$, whose

Algorithm 5 REDUCEMATRIX($\mathcal{D}$)

```
1  $\mathcal{D}' \leftarrow \mathcal{D}$;
2  for $i \leftarrow 1$ to $N+M$ do
3      $r_i \leftarrow \min_{j\in[N+M]} \mathcal{D}'(i,j)$;
4      for $j \leftarrow 1$ to $N+M$ do
5          $\mathcal{D}'(i,j) \leftarrow \mathcal{D}'(i,j) - r_i$;
6  for $j \leftarrow 1$ to $N+M$ do
7      $c_j \leftarrow \min_{i\in[N+M]} \mathcal{D}'(i,j)$;
8      for $i \leftarrow 1$ to $N+M$ do
9          $\mathcal{D}'(i,j) \leftarrow \mathcal{D}'(i,j) - c_i$;
10 $\mathcal{R} \leftarrow \sum_{i=1}^{N+M} r_i + \sum_{j=1}^{N+M} c_j$;
11 return $\mathcal{R}$, $\mathcal{D}'$
```

$(i+1, j+1)$-th entry is the *minimum distance* to travel from the exit of region (or depot) i to the exit of region (or depot) j. Here $i, j \in \{0\} \cup [N+M-1]$. Of note, this region-to-region distance $\mathcal{D}(i+1, j+1)$ includes the length of the path that fully covers region (or depot) j, and it equals to:

$$\mathcal{D}(i+1,j+1) = \begin{cases} \infty, & \text{if } i=j \text{ or } i, j \in \{0\}\cup\{N+1,\dots,N+M-1\} \\ \min_{b_j\in\hat{B}_j} \mathrm{d}(p_0, b_{j1}) + \mathrm{g}(b_j), & \text{if } i=0 \text{ or } i>N \\ \min_{b_i\in\hat{B}_i} \mathrm{d}(b_{in_i}, p_0), & \text{if } j=0 \text{ or } j>N \\ \min_{b_i\in\hat{B}_i, b_j\in\hat{B}_j} \mathrm{d}(b_{in_i}, b_{j1}) + \mathrm{g}(b_j), & \text{else} \end{cases} \tag{19}$$

With this distance matrix, the lower bounding strategy introduced in [56] can be applied to find a lower bound for the original problem (see Algorithm 5). In particular, we reduce each row of $\mathcal{D}$ by subtracting the smallest value in the row from each element of the row (Lines 2-5). Similarly, we also reduce each column of $\mathcal{D}$ by subtracting the smallest value in the column from each element of the column (Lines 6-9). The sum of the reductions constitutes a lower bound and the resulting matrix $\mathcal{D}'$ is called the *reduced matrix*. We place this reduced matrix $\mathcal{D}'$ and associated lower bound at the root node $k=1$, and re-denote them as $\mathcal{D}'_1$ and LB_1, respectively. The lower bounds for subproblems can then be calculated in a similar way. In particular, if the lower bound of subproblem k is LB_k and its reduced matrix is $\mathcal{D}'_k$, the lower bounds and reduced matrices of its successors defined in Eqs. (11)-(12) are then given by

$$LB_{k+1} = LB_k + \mathcal{D}'_k(i_h+1, i_{h+1}+1) + \mathcal{R}_{k+1} \tag{20}$$

$$LB_{k+2} = LB_k + \mathcal{R}_{k+2} \tag{21}$$

$$(\mathcal{R}_{k+1}, \mathcal{D}'_{k+1}) = \text{REDUCEMATRIX}(\mathcal{D}_{k+1}) \tag{22}$$

$$(\mathcal{R}_{k+2}, \mathcal{D}'_{k+2}) = \text{REDUCEMATRIX}(\mathcal{D}_{k+2}) \tag{23}$$

where

$$\mathcal{D}_{k+1}(i,j) = \begin{cases} \infty & \text{if } i=i_h+1 \text{ or } j=i_{h+1}+1 \\ \infty & \text{if } i=i_{h+1}+1 \text{ and } j=i_h+1 \\ \mathcal{D}'_k(i,j) & \text{else} \end{cases}$$

$$\mathcal{D}_{k+2}(i,j) = \begin{cases} \infty & \text{if } i=i_h+1 \text{ and } j=i_{h+1}+1 \\ \mathcal{D}'_k(i,j) & \text{else} \end{cases}$$

Of note, as link (i_h, i_{h+1}) is added to I_{k+1}, $\mathcal{D}_{k+1}(i,j)$ is set to ∞ when $i=i_h+1$ or $j=i_{h+1}+1$ in order to prevent all outgoing links from region i_h and all incoming links to region i_{h+1}. Moreover, when $i=i_h+1$ and $j=i_{h+1}+1$, $\mathcal{D}_{k+1}(i,j)$ is also set to ∞ in order to prevent link (i_{h+1}, i_h). This step ensures that no sub-tours will be constructed.

Nevertheless, the quality of the aforementioned lower bound is not high, as the distance matrix does not reflect the actual region-to-region distances. As we can derive the actual distances between certain pairs of regions from the partial tour $\boldsymbol{\tau}_k$ stored at each node, they can be utilized to improve the lower bound. In particular, starting from the root node, as no information about region-to-region distance can be inferred from $\boldsymbol{\tau}_1$, its lower bound cannot be improved, and hence $LB^*_1 = LB_1$. Now consider the left child of the root node with path $\boldsymbol{\tau}_2 = (p_0, \boldsymbol{b}'_{i_1})$. According to Eq. (19) and Eq. (13), $\mathrm{g}(\boldsymbol{\tau}_2) = \mathcal{D}(1, i_1+1)$, hence there is no update to the distance matrix and $LB^*_2 = LB_2 = LB^*_1 + D'_1(1, i_1+1) + \mathcal{R}_2$. Straightforwardly, $LB^*_3 = LB_3 = LB^*_1 + \mathcal{R}_3$.

Let's next consider the left child of node 2, i.e., node 4 with $\boldsymbol{\tau}_4 = (p_0, \boldsymbol{b}^*_{i_1}, \boldsymbol{b}'_{i_2})$. Note that this path updates two region-to-region distances, depot p_0 to region i_1 and region i_1 to region i_2. Taking these changes into the account, we then have

$$LB^*_4 = LB^*_2 + \mathcal{D}'_2(i_1+1, i_2+1) + \mathcal{R}_4 + \delta_{0i_1} + \delta_{i_1i_2} \tag{24}$$

where

$$\delta_{0i_1} = \left[\mathrm{d}(p_0, b^*_{i_11}) + \mathrm{g}(\boldsymbol{b}^*_{i_1})\right] - \left[\mathrm{d}(p_0, b'_{i_11}) + \mathrm{g}(\boldsymbol{b}'_{i_1})\right] \tag{25}$$

$$\delta_{i_1i_2} = \left[\mathrm{d}(b^*_{i_1n_{i_1}}, b'_{i_21}) + \mathrm{g}(\boldsymbol{b}'_{i_2})\right] - \mathcal{D}(i_1+1, i_2+1) \tag{26}$$

Generally, given the improved lower bound LB^*_k for node k, the improved lower bounds for its successors, node $k+1$ and $k+2$, can be derived as

$$LB^*_{k+1} = LB^*_k + \mathcal{D}'_k(i_h+1, i_{h+1}+1) + \mathcal{R}_{k+1} + \delta_{i_{h-1}i_h} + \delta_{i_hi_{h+1}} \tag{27}$$

$$LB^*_{k+2} = LB^*_k + \mathcal{R}_{k+2} \tag{28}$$

where

$$\delta_{i_{h-1}i_h} = \left[\mathrm{d}(b^*_{i_{h-1}n_{i_{h-1}}}, b^*_{i_h1}) + \mathrm{g}(\boldsymbol{b}^*_{i_h})\right] - \left[\mathrm{d}(b^*_{i_{h-1}n_{i_{h-1}}}, b'_{i_h1}) + \mathrm{g}(\boldsymbol{b}'_{i_h})\right] \tag{29}$$

$$\delta_{i_hi_{h+1}} = \left[\mathrm{d}(b^*_{i_hn_{i_h}}, b'_{i_{h+1}1}) + \mathrm{g}(\boldsymbol{b}'_{i_{h+1}})\right] - \mathcal{D}(i_h+1, i_{h+1}+1) \tag{30}$$

In the special case when $(\boldsymbol{\tau}_{k+1}, p_0)$ constitutes a complete tour, we have

$$\delta_{i_hi_{h+1}} = \left[\mathrm{d}(b^*_{i_hn_{i_h}}, b'_{i_{h+1}1}) + \mathrm{g}(\boldsymbol{b}'_{i_{h+1}}) + \mathrm{d}(b'_{i_{h+1}n_{i_{h+1}}}, p_0)\right] - \left[\mathcal{D}(i_h+1, i_{h+1}+1) + \mathcal{D}(i_{h+1}+1, 1)\right] \tag{31}$$

If this tour is feasible and its length is smaller than the current upper bound, the upper bound is updated.

Having explained the bounding strategy, the branching rule can be described mathematically. In particular, suppose subproblem k is selected for expansion. To generate its successors, we choose i_{h+1} that causes the greatest increase in the lower bound of subproblem k, i.e.,

$$i_{h+1} = \underset{s\in[N+M-1]\backslash\{i_1,i_2,\dots,i_h\}}{\arg\max} \mathcal{R}_s \tag{32}$$

where

$$(\mathcal{R}_s, \mathcal{D}'_s) = \text{REDUCEMATRIX}(\mathcal{D}_s)$$
$$\mathcal{D}_s(i,j) = \begin{cases} \infty & \text{if } i = i_h + 1 \text{ and } j = s+1 \\ \mathcal{D}'_k(i,j) & \text{else} \end{cases} \tag{33}$$

C. Search Strategy

In our BnB method for EMTSP-CPP, we use the depth first search (DFS) as the search strategy, which selects the most recently generated subproblem to be expanded next. Compared with the best first search (BFS), another widely used search strategy that selects the most promising subproblem to be expanded next, the DFS is advantageous in that it requires less storage space for constructing the search tree and can find feasible solutions more quickly. As we will show in the experimental studies, due to these advantages, using DFS significantly reduces the computation time, compared with BFS.

D. BnB Method for EMTSP-CPP

In this subsection, we summarize the key steps of our BnB method to solve problem $\mathcal{P}_1$ (see Algorithm 6). The algorithm starts from finding the set of candidate coverage paths for each region (Lines 1-4) and initializing the search tree (Lines 5-9). It then repeats the following steps until no better feasible solutions can be found:

- **DFS** (Line 11): This step performs depth first search to select the most recently generated subproblem k for expansion.
- **Branching** (Lines 12-14): This step chooses region i_{h+1} that causes the lower bound of subproblem k to increase the most for branching. If more than one region are found, select one at random. With i_{h+1} found, the successors of node k are then generated.
- **Feasibility check** (Lines 15-16): This step checks the feasibility of the constructed path at each newly created node to ensure it meets the energy constraint D_{max}. If the path is feasible and traverses all regions, the upper bound is updated if necessary.
- **Update** (Lines 17-19): This step updates the list of active subproblems, L.

E. Theoretical Analysis

In this subsection, we perform theoretical analysis on the validity of the lower bound calculation and optimality of the final result.

Algorithm 6 BnB Algorithm for EMTSP-CPP $\mathcal{P}_1$

```
Input: $p_0$, $\{P_i\}$, $l$, $w$, $D_{max}$, $M$
Output: tour $\boldsymbol{\tau}^*$, cost $UB$
1 for $i \leftarrow 1$ to $N$ do
2     $\hat{\boldsymbol{B}}_i \leftarrow$BF-CPP($P_i$, $l$, $w$); $c_i \leftarrow$ centroid of region $i$;
3 for $i \leftarrow N+1$ to $N+M-1$ do
4     $\hat{\boldsymbol{B}}_i \leftarrow p_0$; $c_i \leftarrow p_0$;
5 $UB \leftarrow M \times D_{max}$;
6 Construct distance matrix $\mathcal{D}$ using Eq. (19);
7 Create the search tree with a root node 1;
8 Calculate and store the lower bound $LB_1^*$, reduced matrix
  $\mathcal{D}'_1$, and path $\boldsymbol{\tau}_1$ in node 1;
9 $L \leftarrow \{1\}$;
10 while $L \neq \emptyset$ do
11     DFS: Select the most recently generated subproblem
       $k \in L$;
12     Branching: Choose $i_{h+1}$ using Eqs. (32)-(33). If
       multiple regions are found, choose one at random;
13     Generate successors, node $k+1$ and $k+2$;
14     Calculate and store the lower bounds $LB_{k+1}^*$, $LB_{k+2}^*$
       using Eqs. (27)-(31), reduced matrices $\mathcal{D}'_{k+1}$, $\mathcal{D}'_{k+2}$
       using Eqs. (22)-(23), and paths $\boldsymbol{\tau}_{k+1}$, $\boldsymbol{\tau}_{k+2}$ using Eqs.
       (16)-(18) in node $k+1$ and node $k+2$, respectively;
15     Feasibility check: Check the feasibility of paths $\boldsymbol{\tau}_{k+1}$
       and $\boldsymbol{\tau}_{k+2}$. If any of the paths is infeasible, set the
       corresponding lower bound to infinity;
16     If $(\boldsymbol{\tau}_{k+1}, p_0)$ constitutes a complete and feasible tour
       and its length is smaller than $UB$, update $UB$ by
       $UB \leftarrow LB_{k+1}^*$ and set $\boldsymbol{\tau}^* \leftarrow (\boldsymbol{\tau}_{k+1}, p_0)$;
17     Update: Remove $k$ from $L$;
18     Add $k+1$ into $L$, if $LB_{k+1}^* \leq UB$;
19     Add $k+2$ into $L$, if $LB_{k+2}^* \leq UB$;
20 return $\boldsymbol{\tau}^*$, $UB$
```

1) Validity of Lower Bound Calculation: The calculation of the lower bound is based on the concept of reduction and the partially constructed tour $\boldsymbol{\tau}_k$. Let's first prove in Lemma 5.1 that the lower bounds given in Eqs. (20)-(21) are valid. We then show in Theorem 5.2 that the lower bounds given in Eqs. (27)-(31) are also valid. Finally, we show in Theorem 5.3 the convergence of the lower bound.

Lemma 5.1: Let tour $\boldsymbol{t}_k \in \boldsymbol{T}_k$ be a feasible solution to subproblem k with $I_k = \{(0, i_1), (i_1, i_2), \dots, (i_{h-1}, i_h)\}$ and $g(\boldsymbol{t}_k)$ be its length. $\boldsymbol{T}_k$ is the full set of feasible solutions to subproblem k. Then $LB_k \leq g(\boldsymbol{t}_k)$ for all $\boldsymbol{t}_k \in \boldsymbol{T}_k$, where LB_k is obtained using Eqs. (20)-(21).

Proof: Suppose $C(\boldsymbol{t}_k)$ is an estimate of the length of tour $\boldsymbol{t}_k$ calculated using distance matrix $\mathcal{D}$ in Eq. (19), i.e., $C(\boldsymbol{t}_k) = \mathcal{D}(1, i_1+1) + \mathcal{D}(i_{N+M-1}+1, 1) + \sum_{j=1}^{N+M-2} \mathcal{D}(i_j+1, i_{j+1}+1)$. We then have $g(\boldsymbol{t}_k) \geq C(\boldsymbol{t}_k)$ for all $\boldsymbol{t}_k \in \boldsymbol{T}_k$, as each entry of $\mathcal{D}$, $\mathcal{D}(i+1, j+1)$, is the minimum distance to travel from the exit of region i to the exit of region j.

Now consider an asymmetric TSP with distance matrix $\mathcal{D}$. LB_k derived using Eqs. (20)-(21) is then a lower bound for subproblem k, according to Theorems 2-4 in [56]. Therefore, $LB_k \leq C(\boldsymbol{t}_k)$ for all $\boldsymbol{t}_k \in \boldsymbol{T}_k$. Since $C(\boldsymbol{t}_k) \leq \mathrm{g}(\boldsymbol{t}_k)$, we then have $LB_k \leq \mathrm{g}(\boldsymbol{t}_k)$ for all $\boldsymbol{t}_k \in \boldsymbol{T}_k$. The proof is now complete. ■

Theorem 5.2: Let tour $\boldsymbol{t}_k \in \boldsymbol{T}'_k$ be a feasible solution to subproblem k with $I_k = \{(0, i_1), (i_1, i_2), \ldots, (i_{h-1}, i_h)\}$ and $\mathrm{g}(\boldsymbol{t}_k)$ be its length. $\boldsymbol{T}'_k$ is the full set of feasible solutions to subproblem k with $\boldsymbol{b}^*_{i_1}, \boldsymbol{b}^*_{i_2}, \ldots, \boldsymbol{b}^*_{i_{h-1}}$ as the coverage paths for regions $i_1, i_2, \ldots, i_{h-1}$, respectively, where $\boldsymbol{b}^*_{i_r}$, $r \in [h-1]$ is defined by Eq. (16). Then $LB^*_k \leq \mathrm{g}(\boldsymbol{t}_k)$ for all $\boldsymbol{t}_k \in \boldsymbol{T}'_k$, where LB^*_k is obtained using Eqs. (27)-(31).

Proof: Recall that LB^*_k is obtained by correcting region-to-region distances used for computing the lower bound LB_k. When $k = 1, 2, 3$, it is straightforward that $LB^*_k \leq \mathrm{g}(\boldsymbol{t}_k)$ for all $\boldsymbol{t}_k \in \boldsymbol{T}_k$, as $LB^*_k = LB_k \leq \mathrm{g}(\boldsymbol{t}_k)$ for all $\boldsymbol{t}_k \in \boldsymbol{T}_k$ according to Lemma 5.1. When $k = 4$, according to Eqs. (24)-(26), $LB^*_4 = LB_4 + \delta_{0i_1} + \delta_{i_1 i_2}$, where δ_{0i_1} corrects the distance to travel from the depot to the exit of region i_1, using the coverage path $\boldsymbol{b}^*_{i_1}$. As all tours in $\boldsymbol{T}'_4$ adopt $\boldsymbol{b}^*_{i_1}$ as the coverage path for region i_1, the exit location in i_1 must be $\boldsymbol{b}^*_{i_1 n_{i_1}}$. With this information, $\delta_{i_1 i_2}$ further improves the estimate for the minimum distance to travel from the exit of region i_1 to the exit of region i_2. Note that $\delta_{0i_1} \geq 0$ and $\delta_{i_1 i_2} \geq 0$, which can be easily derived according to Eqs. (13)-(15) and Eq. (19). Therefore, essentially, LB^*_4 is a lower bound obtained using distance matrix $\mathcal{D}$ but with entry $\mathcal{D}(1, i_1 + 1)$ replaced by the actual distance to travel from the depot to the exit of region i_1, i.e., $\mathcal{D}(1, i_1 + 1) = \mathrm{d}(p_0, b^*_{i_1 1}) + \mathrm{g}(\boldsymbol{b}^*_{i_1})$, and with entry $\mathcal{D}(i_1 + 1, i_2 + 1)$ replaced by a better estimate of the minimum distance to travel from the exit of region i_1 to the exit of region i_2, i.e., $\mathcal{D}(i_1 + 1, i_2 + 1) = \mathrm{d}(b^*_{i_1 n_{i_1}}, b'_{i_2 1}) + \mathrm{g}(\boldsymbol{b}'_{i_2})$. According to Lemma 5.1, we thus have $LB^*_4 \leq \mathrm{g}(\boldsymbol{t}_4)$ for all $\boldsymbol{t}_4 \in \boldsymbol{T}'_4$.

Similarly, we can derive that in general LB^*_k is essentially a lower bound obtained using distance matrix $\mathcal{D}$ but with entries $\mathcal{D}(1, i_1 + 1), \mathcal{D}(i_1 + 1, i_2 + 1), \ldots, \mathcal{D}(i_{h-2} + 1, i_{h-1} + 1)$ replaced with the actual distance values and $\mathcal{D}(i_{h-1} + 1, i_h + 1)$ replaced with an improved estimate of the minimum distance value. Therefore, $LB^*_k \leq \mathrm{g}(\boldsymbol{t}_k)$ for all $\boldsymbol{t}_k \in \boldsymbol{T}'_k$, according to Lemma 5.1. The proof is now complete. ■

Theorem 5.3: If I_k includes all regions to be covered and $\boldsymbol{\tau} = (\boldsymbol{\tau}_k, p_0)$ is the resulting single tour, then $\mathrm{g}(\boldsymbol{\tau}) = LB^*_k$.

Proof: From Eqs. (16)-(18), we know that $\boldsymbol{\tau} = (p_0, \boldsymbol{b}^*_{i_1}, \boldsymbol{b}^*_{i_2}, \ldots, \boldsymbol{b}^*_{i_{N+M-1}}, p_0)$. From Eqs. (27)-(31), we can derive that LB^*_k is essentially a lower bound obtained using distance matrix $\mathcal{D}$ but with entries $\mathcal{D}(1, i_1 + 1), \mathcal{D}(i_1 + 1, i_2 + 1), \ldots, \mathcal{D}(i_{N+M-2} + 1, i_{N+M-1} + 1), \mathcal{D}(i_{N+M-1} + 1, 1)$ all replaced with the actual distance values. According to Theorem 4 in [56], we then have $\mathrm{g}(\boldsymbol{\tau}) = LB^*_k$. ■

2) Optimality of Final Result: In our BnB method, an approximation is applied to construct the tour progressively where the best coverage path for the current region, given in Eq. (16), is the one that minimizes the distance for traveling from the exit of the previous region, then covering the current region completely and finally arriving at the centroid of the next region. Although our BnB method cannot guarantee of finding a global optima, because of this approximation, it has some nice properties in terms of optimality as shown in the following lemmas and theories.

Lemma 5.4: Suppose $\boldsymbol{\tau} = (\boldsymbol{\tau}_k, p_0)$ at the leaf node k constitutes a complete tour. Then this tour is same as the one derived by applying the FINDTOUR() function, with region visiting order $\boldsymbol{o}$ specified by I_k, i.e., $\boldsymbol{o} = \{i_1, i_2, \ldots, i_{N+M-1}\}$.

Proof: According to Eqs. (16)-(18), $\boldsymbol{\tau} = (p_0, \boldsymbol{b}^*_{i_1}, \boldsymbol{b}^*_{i_2}, \ldots, \boldsymbol{b}^*_{i_{N+M-1}}, p_0)$, where $\boldsymbol{b}^*_{i_r}$, $i_r \in [N + M - 1]$, is same as the coverage path found by applying Line 8 in Algorithm 3. Therefore, $\boldsymbol{\tau}$ is same as the tour generated by the FINDTOUR() function. ■

Lemma 5.5: Assume that the UAV can only choose from the BFPs generated by the BF-CPP algorithm (Algorithm 1) to cover each region. Given the region visiting order $\boldsymbol{o} = \{i_1, i_2, \ldots, i_{N+M-1}\}$, the tour $\boldsymbol{\tau} = (p_0, \boldsymbol{b}^*_{i_1}, \boldsymbol{b}^*_{i_2}, \ldots, \boldsymbol{b}^*_{i_{N+M-1}}, p_0)$ found by the FINDTOUR() function is optimal if for each $r \in [N + M - 2]$,

$$\boldsymbol{b}^*_{i_r} = \underset{\boldsymbol{b}_{i_r}}{\operatorname{argmin}}\ \mathrm{g}(\boldsymbol{b}_{i_r}) + \mathrm{d}(b^*_{i_{r-1} n_{i_{r-1}}}, b_{i_r 1}) + \mathrm{d}(b_{i_r n_{i_r}}, b_{i_{r+1} 1}) \tag{34}$$

is satisfied for any $\boldsymbol{b}_{i_{r+1}} \in \hat{\boldsymbol{B}}_{i_{r+1}}$, where $b^*_{i_{r-1} n_{i_{r-1}}} = p_0$ when $r = 1$.

Proof: The optimality of tour $\boldsymbol{\tau}$ can be proved by inspecting each region in the order specified by $\boldsymbol{o}$. In particular, starting from depot p_0, condition (34) tells that $\boldsymbol{b}^*_{i_1}$ is the optimal path to cover region i_1 as it leads to the shortest distance to travel from p_0 to region i_1, fully cover this region i_1 and then reach the next region i_2, no matter from where to exit region i_2. With $\boldsymbol{b}^*_{i_1}$ determined as the optimal coverage path for region i_1, we then regard the exit location $b^*_{i_1 n_{i_1}}$ as the start location and prove the optimality of partial tour $(b^*_{i_1 n_{i_1}}, \boldsymbol{b}^*_{i_2}, \ldots, \boldsymbol{b}^*_{i_{N+M-1}}, p_0)$. By repeating the above procedure, we can prove the optimality of each coverage path $\boldsymbol{b}^*_{i_r}$, and thus the optimality of the complete tour $\boldsymbol{\tau}$. ■

Theorem 5.6: Let $\boldsymbol{\tau}^*$ be the optimal solution to problem $\mathcal{P}_1$ and $\boldsymbol{o}^*$ be the corresponding region visiting order. Assume that the UAV can only choose from the BFPs generated by the BF-CPP algorithm (Algorithm 1) to cover each region. If given $\boldsymbol{o}^*$, the tour found by applying function FINDTOUR() is optimal, then the solution returned by the BnB method in Algorithm 6 is optimal.

Proof: Suppose $\boldsymbol{o}^* = \{i^*_1, i^*_2, \ldots, i^*_{N+M-1}\}$. If a leaf node k has $I_k = \{(0, i^*_1), (i^*_1, i^*_2), \ldots, (i^*_{N+M-2}, i^*_{N+M-1})\}$, then $(\boldsymbol{\tau}_k, p_0)$ constitutes a complete tour and $\boldsymbol{\tau}_k = (p_0, \boldsymbol{b}^*_{i^*_1}, \boldsymbol{b}^*_{i^*_2}, \ldots, \boldsymbol{b}^*_{i^*_{N+M-1}})$. According to Lemma 5.5, our BnB method finds the same tour as the one obtained by FINDTOUR(), given a region visiting order. In addition, if the tour found by FINDTOUR() is optimal, then the coverage path found for each region must also be optimal. Therefore, $\boldsymbol{b}^*_{i^*_r}$ is optimal for each $r \in [N + M - 1]$, and hence $(\boldsymbol{\tau}_k, p_0) = \boldsymbol{\tau}^*$.

Next we prove that our BnB method will generate such leaf node k and return the optimal tour $\boldsymbol{\tau}^*$. This can be proved by showing that for each node s with $I_s \subseteq I_k$, $LB^*_s \leq UB$, which ensures that node s will be expanded in certain iteration and thus the leaf node k will be finally generated. To prove this, let $I_s = \{(0, i^*_1), (i^*_1, i^*_2), \ldots, (i^*_{h-1}, i^*_h)\}$, where $h \in [N + M - 1]$ and $i^*_{h-1} = 0$ if $h = 1$. According to Theorem 5.2, $LB^*_s \leq \mathrm{g}(\boldsymbol{t}_s)$ for all $\boldsymbol{t}_s \in \boldsymbol{T}'_s$, where $\boldsymbol{T}'_s$ is the full set of

$$\left\{ \left\{\boxed{1},\boxed{3},\boxed{5}\right\},\ \left\{\boxed{2},\boxed{6}\right\},\ \left\{\boxed{4},\boxed{7},\boxed{8},\boxed{9}\right\} \right\}$$

Fig. 3. Sets-based chromosome representation.

feasible solutions to subproblem s with the coverage paths for regions $i_1^*, i_2^*, \ldots, i_{h-1}^*$ being $b_{i_1^*}^*, b_{i_2^*}^*, \ldots, b_{i_{h-1}^*}^*$, respectively. As $\tau^* \in T_s'$, we have $LB_s^* \leq g(\tau^*)$. Furthermore, since τ^* is the optimal solution, we have $g(\tau^*) \leq UB$. Therefore, $LB_s^* \leq UB$ holds for all s with $I_s \subseteq I_k$. ■

Remark 1: In Lemma 5.4, Eq. (34) is a sufficient condition for the optimality of the tour generated by the FINDTOUR() function and it can be satisfied when the size of the next region i_{r+1} is small or region i_{r+1} is far away from the current region i_r. However, it is not a necessary condition. Actually, as we will show in the experimental results, our BnB method that essentially applies the FINDTOUR() function to construct tours generates optimal solutions to all problems we have tested and can verify.

VI. Genetic Algorithm for Solving $\mathcal{P}_1$ and $\mathcal{P}_2$

The BnB method introduced in the previous section is able to solve problem $\mathcal{P}_1$ with high accuracy. Nevertheless, it cannot solve problem $\mathcal{P}_2$ that aims to balance the workload among UAVs, and the problem scale it can handle is limited. To address these limitations, in this section, we introduce a genetic algorithm (GA) that can solve large-scale EMTSP-CPP, both $\mathcal{P}_1$ and $\mathcal{P}_2$, efficiently.

The key of our GA for EMTSP-CPP is to properly assign regions to each UAV. Based on a region assignment, the original EMTSP-CPP can be decomposed into a group of energy constrained TSP-CPP with a single UAV. Each energy constrained TSP-CPP can be solved by applying the Fast NN-2Opt algorithm first, and penalizing the violation of energy constraint later in the fitness function. The main features of our GA for EMTSP-CPP are described as follows.

A. Chromosome Design

The chromosome design is the key to the performance of a GA. A good chromosome design should have nearly no redundancy in representing the solutions, which means a solution should avoid having multiple representations. Inspired by the grouping GA introduced in [60] for MTSP, we propose a sets-based chromosome representation, where each chromosome consists of M sets of regions and each set is assigned to a UAV. There is no ordering among the regions in each set as well as among the sets. An illustration of the sets-based chromosome representation is shown in Fig. 3, which represents an assignment of $N = 9$ regions to $M = 3$ UAVs. Note that the depot is not included in the chromosome, as each tour starts and ends at the depot by default.

Our sets-based chromosome design brings two advantages over the traditional chromosome designs for MTSP [60]–[63], which specify both the target location assignment and the visiting order. First, our sets-based design only represents the region assignment, which leads to a much smaller search space, and thus the algorithm is expected to converge much faster. Second, our design avoids having redundancy in representing the tours, which are inherent in the traditional designs for MTSP. Meanwhile, there is an inherent drawback in such a sets-based chromosome design. With less control information represented by the chromosome, it increases the computation burden for following steps. In particular, to compute the fitness of a chromosome, we need to first generate tours based on the region assignment specified by the chromosome, which makes the evolution of each generation to be costly. However, as we will show in the experimental results, the advantages of this sets-based chromosome outweigh its disadvantage.

B. Fitness

As our sets-based chromosomes only specify region assignments, in order to evaluate the fitness of a chromosome, we first apply the Fast NN-2Opt algorithm introduced in Section IV-B to generate tours based on the chromosome. The cost of the derived tours is then measured as the fitness of the chromosome, using J_1 if $\mathcal{P}_1$ is considered or J_2 if $\mathcal{P}_2$ is considered. However, due to energy constraints, some tours may be infeasible. To address this issue, instead of directly removing infeasible solutions, which may cause the search to "stagnate" when the number of feasible solutions is small [64], we here employ a penalty factor to penalize infeasible tours. The fitness of a chromosome is then defined as the sum of the cost of the resulting tours and the penalty factor $\lambda \geq 0$, i.e.,

$$\text{Fitness} = J_i + \lambda \tag{35}$$

where $i \in \{1, 2\}$.

Different types of penalties have been developed to handle constraints in evolutionary algorithms [65]. We here adopt the following dynamic penalty function [65], which increases over time as the generation number t grows:

$$\lambda = (Ct)^{\alpha} \sum_{i=1}^{m} (\max\{0, d_i(t) - D_{max}\})^{\beta}$$

In the above equation, C, α, and β are constants. $d_i(t)$, $i \in [m]$ is the cost of the ith tour obtained at the tth generation. Note that $\lambda = 0$ when the chromosome results in M feasible tours.

C. Crossover

Given the parents, a proper crossover operator is required to produce off-springs. Considering the unique features of the sets-based chromosomes, we design a new crossover operator based on the idea of ordered crossover [66] and the one introduced in [60]. In particular, given two parents, labeled as A and B, our crossover operator first selects the most promising set from parent A and copies this set to the child. It then deletes all regions belonging to this set from the sets of parent B. After that, the operator examines the remaining sets of parent B. If there are exactly $M - 1$ nonempty sets, these sets are directly copied to the child. Otherwise, the operator adjusts the regions in the sets to produce $M - 1$ nonempty sets and then copies these sets to the child. Specifically, if there are fewer than $M-1$ nonempty sets, the operator picks a region from the largest set uniformly at random and moves this region to an empty set. This step is repeated until there are exactly $M-1$ nonempty sets. On the other hand, if there are M nonempty sets, the operator picks the smallest set and moves all regions from this set to the other sets one by one, and at each move, a region is moved to the second smallest set. Finally, by switching the role of parent A and parent B, the second child can be generated using the same procedure.

In the above procedure, the most promising set is defined in different ways for solving EMTSP-CPP with different objectives. For problem $\mathcal{P}_1$ that aims to minimize the total travel distance, the most promising set is the one that leads to a feasible tour with the smallest ratio of tour length to the number of regions in the tour. If all sets result in infeasible tours, the most promising set is selected solely based on the ratio. For problem $\mathcal{P}_2$ that aims to minimize the longest tour, the most promising set is the one that leads to the shortest tour.

D. Mutation

The mutation is often applied with a low probability, which helps introduce randomness within the chromosomes to increase the coverage of the search space. For a specific chromosome, two regions are randomly picked separately from two randomly selected sets, then they are interchanged.

E. Initial Population Generation

The initial population is generated using simple greedy heuristics inspired by the approach introduced in [63], and each set in a chromosome is generated with the region visiting order considered at the same time. For EMTSP-CPP with different objectives, we use different heuristics to generate the initial population. In particular, for problem $\mathcal{P}_1$ that aims to minimize the total travel distance, we first initialize the M sets by inserting a randomly selected region to each set, which ensures that no sets will be empty. After that, for the remaining $N-M$ unassigned regions, we find the region that is the closest to the last region in any of the M sets and insert this region to the end of the set closest to it, where the distance between two regions is approximated using regions' centroids. To reduce the chance of generating infeasible tours, after inserting a region into a set, we approximate the length of the resulting tour, and stop assigning more regions into the set if the tour is long enough. This process is repeated until there are no unassigned regions.

For problem $\mathcal{P}_2$ that aims to minimize the longest tour, the regions are assigned to the sets in a round-robin fashion. In particular, after initializing each set with a randomly selected region, we insert an unassigned region to the end of each set in turn, where the selected region is the one that is the closest to the last region in the set. Similarly, we examine the updated set at each iteration to reduce the chance of generating infeasible tours and continue this process until all regions are assigned.

TABLE I
TOTAL TRAVEL DISTANCE OF ALL UAVS AND MEAN COMPUTATIONAL TIME FOR BNB AND DP ON DIFFERENT TEST PROBLEMS

Instance		Cost (J_1)			Time (s)	
num	N	**BnB**	**DP**	**error**	**BnB**	**DP**
1	2	836.7	836.7	0	0.0272	0.0443
2	3	929.9	929.9	0	0.0327	0.0532
3	4	1119.1	1119.1	0	0.0399	0.1254
4	5	1373.3	1373.7	0	0.0502	0.3521
5	6	1208.5	1208.5	0	0.0824	1.3292
6	7	1484.0	1484.0	0	0.1387	4.4375
7	8	1693.5	1693.5	0	0.2337	21.236
8	9	1805.6	1805.6	0	1.1937	93.746
9	10	1954.8	1954.8	0	2.1056	472.54
10	11	1933.3	1933.3	0	1.9154	2775.4
11	12	2236.0	2236.0	0	19.597	11458

F. Other Features

In our GA, the parents are selected using the typical rank selection method [67], which first ranks the individuals in a population based on their fitness and then picks two parents based on the ranks. We also employ Elitism when updating the population, which propagates a small set of fittest members of the current population to the next generation. This ensures that the quality of the solution derived by the GA does not degrade from one generation to the next.

VII. COMPUTATIONAL EXPERIMENTS

In this section, we conduct computational experiments to evaluate the performance of the proposed approaches for EMTSP-CPP under different objectives. All approaches are implemented in MATLAB and experiments are run on a MacBook Pro with a 2.6GHz 6-Core Intel Core i7 processor and 16GB memory.

A. Performance of the BnB Method

In this study, we investigate the performance of the proposed BnB method in terms of optimality and efficiency. The impact of the search strategy is also exploited.

1) Optimality Study: In Section V-E.2, we have analyzed the optimality of the final result derived by our BnB method from the theoretical point of view. Here, we conduct computational experiments to demonstrate its optimality. As there are no exact approaches available in the literature that can be directly used to solve EMTSP-CPP within reasonable time, we run the dynamic programming (DP)-based approach for TSP-CPP as the benchmark, which is proved to generate optimal solutions [26]. As EMTSP-CPP is a generalization of TSP-CPP, our BnB method can be directly used to solve TSP-CPP.

Table I summarizes the solutions obtained by the two methods to different TSP-CPP with the number of UAVs set to $M=1$ and the maximum distance each UAV can travel set to $D_{max}=\infty$. The regions are randomly generated convex polygons with a random shape and size, and are randomly distributed over the space without overlap. The sensing range of the UAV is set to 1.5×3. To reduce experimental uncertainties, each experiment is repeated for 10 times and the mean values are recorded.

TABLE II

TOTAL TRAVEL DISTANCE OF ALL UAVs AND MEAN COMPUTATIONAL TIME FOR BnB WITH DFS OR BFS ON DIFFERENT TEST PROBLEMS

Instance				Cost (J_1)		Time (s)	
num	N	M	D_{max}	**DFS**	**BFS**	**DFS**	**BFS**
1	5	1	∞	1373.7	1373.7	0.0502	0.0532
2	5	2	∞	1431.6	1431.6	0.1244	0.1324
3	5	2	1200	1458.5	1458.5	0.1207	0.1314
4	5	2	1000	1459.0	1459.0	0.1154	0.1245
5	5	2	800	1545.8	1545.8	0.1240	0.1712
6	5	3	∞	1516.4	1516.4	0.2052	0.2396
7	5	3	800	1554.7	1554.7	0.2627	0.3372
8	8	1	∞	1693.5	1693.5	0.2337	0.2708
9	8	2	∞	1734.3	1734.3	2.7541	11.815
10	8	2	1200	1830.0	1830.0	5.3925	42.905
11	8	2	1000	1840.6	1840.6	3.5176	15.933
12	8	3	∞	1802.6	1802.6	13.574	464.23
13	8	3	1200	1849.6	1849.6	22.527	2797.0
14	8	3	800	1870.8	1870.8	18.002	846.93
15	10	1	∞	1954.8	1954.8	2.1056	7.2192
16	10	2	∞	2005.4	2005.4	24.362	1227.3
17	10	2	1400	2053.3	2053.3	90.481	5178.5
18	10	2	1200	2057.7	2057.7	55.406	4445.6
19	10	3	∞	2067.8	Not	159.89	Not
20	10	3	1000	2108.3	Not	197.03	Not

As we can see, our BnB finds optimal solutions to all the test problems, even though its optimality can only be proved theoretically under certain assumptions.

2) Efficiency Study: Table I also shows the mean computational times of the two methods, both of which increase exponentially with the increase of the number of regions N, as TSP-CPP is an NP-hard problem. Nevertheless, we can see that BnB is much more efficient than DP.

3) Impact of Search Strategy: BFS is another popular search strategy that has been frequently used in BnB for solving TSP/MTSP. Here, we investigate the impact of search strategy on the performance of the proposed BnB method for solving EMTSP-CPP. Table II shows the performance of different search strategies in solving different test problems. Each value in the table is obtained by averaging the results from 10 repeated experiments. The time limit for both algorithms is set to 10,800 seconds (i.e., 3 hours). When no feasible solution is found, we report 'Not'.

The results show the efficiency of the DFS strategy in solving EMTSP-CPP ($\mathcal{P}_1$). We also note that the parameters N, M and D_{max} have the same impact on the efficiency of the two methods. Among them, N and M control the problem scale and increasing any of them will cause more computational time. D_{max} controls the size of the feasible solution space. Interestingly, decreasing D_{max} sometimes increases the computational time, but sometimes reduces the computational time. This is because, on one hand, smaller D_{max} reduces the feasible solution space, thus making it harder to find feasible solutions. On the other hand, a tighter constraint causes many nodes to fail the feasibility check and thus be pruned early.

B. Performance of the GA

In this section, we conduct experiments to evaluate the performance of the proposed GA for EMTSP-CPP. We first introduce the benchmark methods implemented in this study for comparison. We then show the experimental results.

TABLE III

TOTAL TRAVEL DISTANCE OF ALL UAVs AND THE COMPUTATIONAL TIME EVALUATED FOR BnB, GASC AND GA2PC ON DIFFERENT TEST PROBLEMS, WHEN THE TWO GAs ADOPT THE FIRST TERMINATION CRITERIA

Instance				Cost (J_1)			Time (s)
num	N	M	D_{max}	**BnB**	**GASC**	**GA2PC**	
1	6	2	∞	**1260.8**	**1260.8**	**1275.1**	0.3042
2	6	2	1000	1275.1	1275.1	1275.1	0.3692
3	6	3	∞	1346.6	1346.6	1346.6	0.8308
4	6	3	600	1418.8	1418.8	1418.8	3.4603
5	6	4	∞	1448.9	1448.9	1448.9	3.5806
6	6	4	600	1478.1	1478.1	1478.1	3.8872
7	6	4	500	1518.7	1518.7	1518.7	5.1808
8	8	2	∞	1734.3	1734.3	1734.3	2.7541
9	8	2	1200	1830.0	1830.0	1830.0	5.3925
10	8	2	1000	1840.6	1840.6	1840.6	3.5176
11	8	3	∞	1802.6	1802.6	1802.6	13.574
12	8	3	1200	1849.6	1849.6	1849.6	22.527
13	8	3	800	1870.8	1870.8	1870.8	18.002
14	8	4	∞	1917.9	1917.9	1917.9	74.2718
15	8	4	800	1939.1	1939.1	1939.1	91.2427
16	10	2	∞	**2005.4**	**2005.4**	**2053.3**	24.362
17	10	2	1400	**2053.3**	**2058.3**	**2071.9**	90.481
18	10	2	1200	2057.7	2057.7	2057.7	55.407
19	10	3	∞	**2067.8**	**2067.8**	**2112.7**	159.89
20	10	3	1000	2108.3	2108.3	2108.3	197.03

TABLE IV

TOTAL TRAVEL DISTANCE OF ALL UAVs AND COMPUTATIONAL TIME EVALUATED FOR BnB, GASC AND GA2PC ON DIFFERENT TEST PROBLEMS, WHEN THE TWO GAs ADOPT THE SECOND TERMINATION CRITERIA

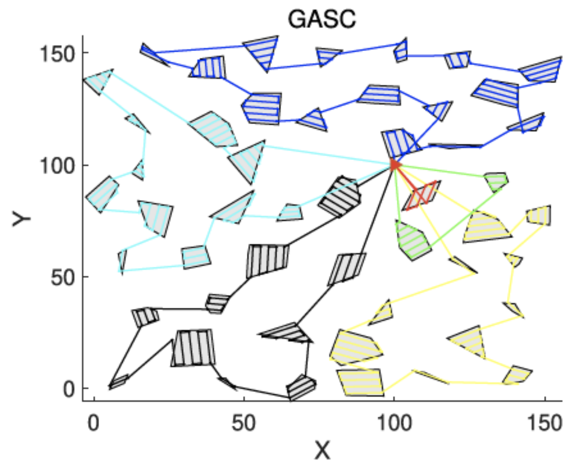
Instance				Cost (J_1)			Time (s)		
num	N	M	D_{max}	**BnB**	**GASC**	**GA2PC**	**BnB**	**GASC**	**GA2PC**
1	6	2	∞	**1260.8**	**1275.1**	**1275.1**	0.3042	0.2271	0.1833
2	6	2	1000	1275.1	1275.1	1275.1	0.3692	0.3413	0.2699
3	6	3	∞	1346.6	1346.6	1346.6	0.8308	0.3702	0.2750
4	6	3	600	1418.8	1418.8	1418.8	3.4603	0.3697	0.1699
5	6	4	∞	1448.9	1448.9	1448.9	3.5806	0.3508	0.2770
6	6	4	600	1478.1	1478.1	1478.1	3.8872	0.2064	0.2966
7	6	4	500	1518.7	1518.7	1518.7	5.1808	0.3646	0.1988
8	8	2	∞	1734.3	1734.3	1734.3	2.7541	0.2711	0.2073
9	8	2	1200	1830.0	1830.0	1830.0	5.3925	0.3135	0.2545
10	8	2	1000	1840.6	1840.6	1840.6	3.5176	0.2665	0.2050
11	8	3	∞	1802.6	1802.6	1802.6	13.574	0.2658	0.2102
12	8	3	1200	1849.6	1849.6	1849.6	22.527	0.2931	0.2295
13	8	3	800	1870.8	1870.8	1870.8	18.002	0.2370	0.2076
14	8	4	∞	1917.9	1917.9	1917.9	74.2718	0.2557	0.1946
15	8	4	800	1939.1	1939.1	1939.1	91.2427	0.2663	0.3262
16	10	2	∞	**2005.4**	**2005.4**	**2060.2**	24.362	0.3906	0.2234
17	10	2	1400	**2053.3**	**2058.3**	**2066.0**	90.481	0.3251	0.2280
18	10	2	1200	2057.7	2057.7	2057.7	55.407	0.3164	0.2426
19	10	3	∞	**2067.8**	**2108.3**	**2108.3**	159.89	0.3203	0.2345
20	10	3	1000	**2108.3**	**2108.3**	**2110.8**	197.03	0.3760	0.2653

1) Benchmark Methods: To evaluate the performance of the proposed GA, we choose two methods as the benchmarks. One is the proposed BnB, which can find (near) optimal solutions to small-scale EMTSP-CPP. Another one is a two-part chromosome based GA that adopts the two-part chromosome representation proposed in [63]. This method can help us understand the performance of the proposed GA in solving large-scale problems, as well as the effectiveness of the proposed new GA features.

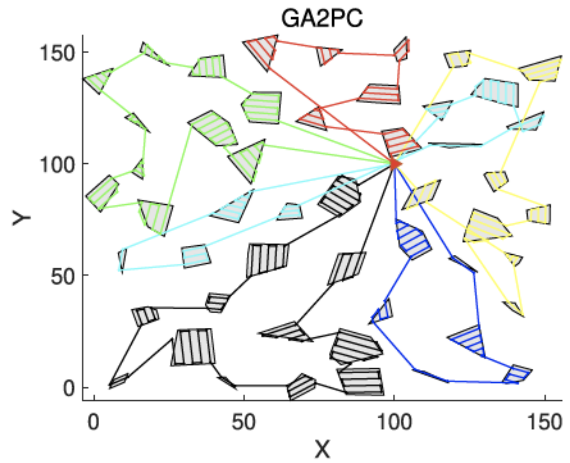
TABLE V

TOTAL TRAVEL DISTANCE OF ALL UAVS AND THE NUMBER OF GENERATIONS EVALUATED FOR GASC AND GA2PC ON DIFFERENT TEST PROBLEMS

Instance				Cost (J_1)		# generations	
num	N	M	D_{max}	GASC	GA2PC	GASC	GA2PC
1	20	5	∞	3778.2	3780.5	281	1476
2	20	5	1000	3909.0	3962.1	302	1465
3	20	5	900	3926.5	3971.2	284	1445
4	20	8	∞	4112.3	4209.5	290	1378
5	20	8	1000	4137.9	4209.5	322	1368
6	20	8	800	4260.6	4332.1	314	1415
7	20	10	∞	4445.2	4457.3	349	1318
8	20	10	1000	4422.4	4483.2	412	1319
9	20	10	800	4492.2	4524.5	348	1434
10	50	5	∞	2671.7	3023.8	23	695
11	50	5	900	3046.1	3204.8	23	695
12	50	5	700	3073.2	Not	38	727
13	50	8	∞	3124.4	3495.6	35	754
14	50	8	550	3821.0	Not	58	726
15	50	10	∞	3489.3	3717.1	45	720
16	50	10	700	3778.5	4286.4	58	706
17	50	10	500	4296.8	Not	60	708
18	50	20	∞	5362.9	5473.5	65	497
19	50	20	800	5400.2	5766.5	71	562
20	50	20	600	5576.9	5939.7	77	591
21	100	5	∞	8715.0	11746	5	320
22	100	5	2200	9640.0	Not	12	373
23	100	8	∞	9445.7	12938	10	365
24	100	8	2000	10824	Not	13	352
25	100	10	∞	10403	13468	8	359
26	100	10	2000	11484	Not	13	326
27	100	10	1400	12320	Not	23	326
28	100	20	∞	14452	17380	10	348
29	100	20	2000	14895	18596	21	314
30	100	20	1200	16175	Not	28	301



(a)



(b)

Fig. 4. Visualization of the tours found by a) GASC and b) GA2PC with the objective to minimize the total travel cost, when $N = 50$, $M = 6$ and $D_{max} = 1000$. Shaded grey areas and the red triangle represent regions and the depot, respectively. Tours assigned to different UAVs are highlighted by different colors.

In the two-part chromosome based GA, the chromosome specifies not only the region assignment, but also the region visiting order for each assignment. In particular, each chromosome consists of two parts, with the first part of length N being a permutation of the N regions and the second part of length M specifying the number of cities to be visited by each UAV. Combining these two parts, we can derive the set of regions to be visited by each UAV as well as the associated region visiting order. Given this information, we can then apply the FINDTOUR() function in Algorithm 3 to generate the tour. To produce new offspring, this two-part chromosome based GA adopts the same crossover operators used in [63]. Specifically, the ordered crossover is used for the first part of the chromosome and the single point asexual crossover is used for the second part. The mutation is achieved using the swap mutation method and the other steps are the same as the proposed GA.

2) Computational Results for Solving $\mathcal{P}_1$: In this subsection, we consider the objective of minimizing the total travel distance (i.e., $\mathcal{P}_1$). Two studies of different scales are conducted.

Small-Scale Study: In the small-scale study, we consider problems with the number of regions $N \leq 10$, and compare the proposed GA with both benchmark methods. The population size, mutation probability and elitism rate of the two GAs are set to 10, 0.05 and 0.1, respectively. The parameters of the dynamic penalty function are set to $C = 0.5$, $\alpha = 1$ and $\beta = 2$. All these parameter values are chosen empirically. Regarding the termination criteria, to better understand the performance of the proposed GA, we consider two termination criteria. The first one makes the GAs terminate after they have run approximately the same amount time as required by the BnB method, the results of which are shown in Table III. The second one makes the GAs terminate when the best fitness score is unchanged for 10 successive generations, the results of which are shown in Table IV. Each experiment is repeated for 10 times and the best solution is recorded. In the tables, GASC refers to the proposed GA that uses the sets-based chromosome and GA2PC refers to the benchmark GA that employs the two-part chromosome. To facilitate analysis, cost values are highlighted in bold when the solutions generated by the three methods have different costs.

As shown in Table III, which presents the results obtained by running each algorithm for roughly the same amount of time, both GAs can find optimal solutions when the problem size is small, but they cannot guarantee it. Moreover, the proposed GA generates equally good or better solutions than

TABLE VI
LENGTH OF THE LONGEST TOUR AND THE NUMBER OF GENERATIONS EVALUATED FOR GASC AND GA2PC ON DIFFERENT TEST PROBLEMS

Instance				Cost (J_2)		# generations	
num	N	M	D_{max}	**GASC**	**GA2PC**	**GASC**	**GA2PC**
1	20	5	∞	852.7	852.7	295	1621
2	20	5	1000	852.7	866.5	295	1541
3	20	5	900	848.0	848.0	339	1579
4	20	8	∞	610.9	610.9	348	1505
5	20	8	1000	610.9	610.9	338	1497
6	20	8	800	610.9	610.9	328	1453
7	20	10	∞	592.7	592.7	355	1375
8	20	10	1000	592.7	592.7	354	1388
9	20	10	800	592.7	592.7	342	1362
10	50	5	∞	652.0	702.7	43	751
11	50	5	900	650.5	733.1	75	744
12	50	5	700	648.5	Not	50	745
13	50	8	∞	534.6	559.3	62	742
14	50	8	550	530.7	Not	69	713
15	50	10	∞	**499.0**	**496.0**	102	711
16	50	10	700	490.2	503.3	89	705
17	50	10	500	**493.3**	**490.5**	66	560
18	50	20	∞	428.4	428.4	131	594
19	50	20	800	428.4	428.4	147	609
20	50	20	600	428.4	428.4	141	626
21	100	5	∞	1979.4	2631.7	9	364
22	100	5	2200	2082.1	Not	11	348
23	100	8	∞	1484.9	2024.0	16	388
24	100	8	2000	1494.4	Not	17	380
25	100	10	∞	1328.3	1592.4	19	358
26	100	10	2000	1338.5	1620.3	16	332
27	100	10	1400	1348.5	Not	23	280
28	100	20	∞	1043.2	1102.1	41	321
29	100	20	2000	1045.7	1091.1	42	316
30	100	20	1200	1041.5	1098.8	44	365

the benchmark GA in all test problems. Now let's analyze Table IV, which shows the results obtained when the GAs cannot find better solutions in 10 consecutive generations. Under this termination criteria, both GAs stop running after a short amount of time, and the proposed GA takes more time to terminate, as it requires more time to evaluate each generation. In addition, we can observe that both GAs can find optimal solutions when the problem size is small, but their performance degrades when the problem size increases and such degradation is obvious for the benchmark GA. Furthermore, by comparing Table IV and Table III, we can observe that, although the running time is significantly reduced, the quality of the solution found by the proposed GA does not decline in most cases.

Large-Scale Study: In the large-scale study, we consider problems with the number of regions $N \geq 20$. The proposed GA is only compared with the two-parts chromosome based GA, as BnB cannot solve these problems within tolerable time limits. In this experiment, we set the population size of both GAs to 100. The other parameters are kept the same. For fair comparison, each algorithm runs for 1 minute to solve each test problem and the best solution obtained from 10 experimental runs is recorded. The results are summarized in Table V, which also shows the number of generations each algorithm evaluated within 1 minute. For illustration purpose, we also visualize in Fig. 4 the tours generated by the two GAs when $N = 50$, $M = 6$ and $D_{max} = 1000$.

As shown in Table V, the proposed GA generates shorter tours than the benchmark GA in all test problems, despite the fact that it takes much more time to generate and examine one generation, as indicated by the smaller number of generations it is able to evaluate within 1 minute. For some test problems, the benchmark GA even fails to generate feasible solutions, such as the test problem 12.

We can also observe from Table V that the total travel distance increases with the increase of the number of UAVs M, when N and D_{max} are fixed. This is because the inclusion of each additional UAV increases the number of depot-region links by two, as each UAV has to depart from and return to the same depot. Another observation we can obtain from the table is that the total travel distance generally increases with the decrease of D_{max}, when N and M are fixed. This is because smaller D_{max} reduces the feasible solution space, causing some solutions that are feasible at larger D_{max} to become infeasible.

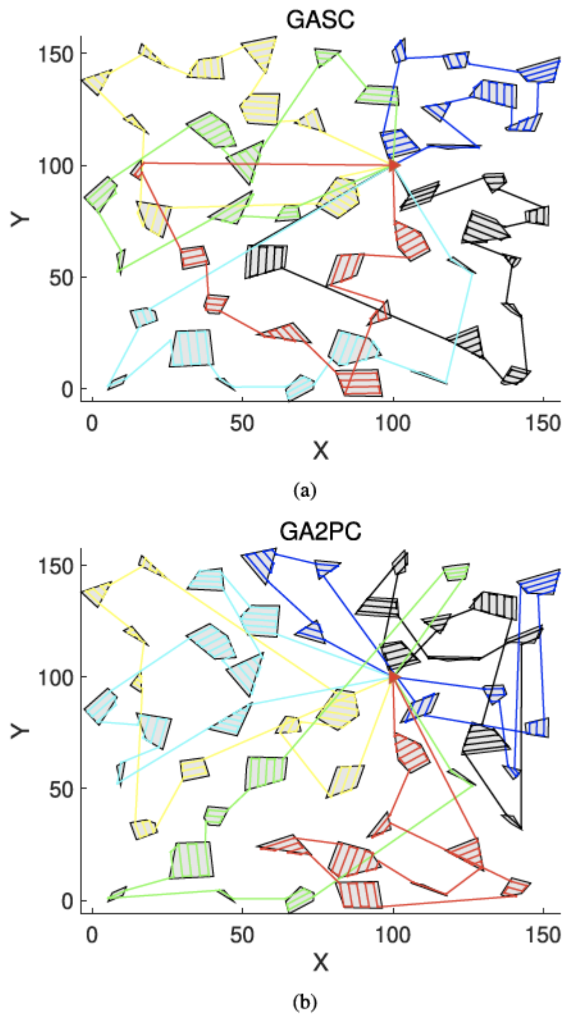


Fig. 5. Visualization of the tours found by a) GASC and b) GA2PC with the objective to minimize the longest tour, when $N = 50$, $M = 6$, and $D_{max} = 1000$.

C. Computational Results for Solving $\mathcal{P}_2$

In this subsection, we consider the objective of minimizing the longest tour (i.e., $\mathcal{P}_2$). In this experiment, the proposed GA is compared only with the two-part chromosome based GA, as the BnB method cannot solve $\mathcal{P}_2$. The same experimental setting for the large-scale study described in the previous subsection is adopted here to configure the parameters of the two GAs. The results are summarized in Table VI. To facilitate analysis, cost values are highlighted in bold when the benchmark GA outperforms the proposed GA. As an illustration, Fig. 5 visualizes the tours generated by the two GAs when $N = 50$, $M = 6$, and $D_{max} = 1000$.

As shown in Table VI, the proposed GA still outperforms the benchmark GA in most test problems, and its advantage becomes more obvious when the problem scale increases. Of interest, the length of the longest tour decreases with the increase of the number of UAVs M, when N and D_{max} are fixed. This is because with more UAVs to share the workload, fewer regions are assigned to each UAV, thus leading to shorter tours. Moreover, we can observe that the value of D_{max} does not impact the length of the longest tour much, when N and M are fixed. This is because D_{max} just restricts the maximum distance a UAV can fly. When the objective is to balance the workload, D_{max} does not help in searching for shorter tours, as long as the tours meet the energy constraints.

VIII. Conclusion

This paper addresses a new multi-UAV path planning problem, called EMTSP-CPP, which aims to find the optimal tours for a set of UAVs with limited power supply to fully cover multiple non-overlapping convex polygonal regions. To solve this problem, a BnB method was first developed. This method adopts the DFS strategy and creates a binary search tree to find the best region visiting order, with tours constructed progressively during the search. Theoretical analyses reveal the optimality of this method. In order to solve large-scale EMTSP-CPP efficiently, we further developed a new GA, which is featured by a new sets-based chromosome representation, new crossover and mutation operators, and a constraint-aware fitness function. Comprehensive computational studies demonstrated the promising performance of the proposed approaches from different aspects. Particularly, the proposed BnB method can generate (near) optimal solutions and is more efficient than the DP-based method. The studies on the impact of search strategy demonstrated the efficiency of the DFS strategy compared with the BFS strategy. Moreover, the proposed GA achieves a good tradeoff between optimality and efficiency, compared with the BnB method. It also outperforms the two-part chromosome based GA in both optimality and efficiency. In the future, we will conduct sensitivity study to understand the impact of the parameters in GA. We will also explore efficient MILP formulations for EMTSP-CPP and its real applications.

References

[1] L. Lin and M. A. Goodrich, "UAV intelligent path planning for wilderness search and rescue," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 709–714.

[2] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "UAVs for smart cities: Opportunities and challenges," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, May 2014, pp. 267–273.

[3] 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.

[4] S. M. Adams and C. J. Friedland, "A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management," in *Proc. 9th Int. Workshop Remote Sens. Disaster Response*, vol. 8, 2011, pp. 12–19.

[5] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, 2013.

[6] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 2, pp. 231–247, Jun. 1992.

[7] 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, Aug. 2012.

[8] M. Dror, A. Efrat, A. Lubiw, and J. S. B. Mitchell, "Touring a sequence of polygons," in *Proc. 35th ACM Symp. Theory Comput. (STOC)*, 2003, pp. 473–482.

[9] K. Obermeyer, "Path planning for a UAV performing reconnaissance of static ground targets in terrain," in *Proc. AIAA Guid., Navigat., Control Conf.*, Aug. 2009, p. 5888.

[10] C. Wei-Pang and S. Ntafos, "The zookeeper route problem," *Inf. Sci.*, vol. 63, no. 3, pp. 245–259, Sep. 1992.

[11] X. Tan and T. Hirata, "Finding shortest safari routes in simple polygons," *Inf. Process. Lett.*, vol. 87, no. 4, pp. 179–186, Aug. 2003.

[12] T. Cabreira, L. Brisolara, and P. R. Ferreira, Jr., "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, Jan. 2019.

[13] T. Bektas, "The multiple traveling salesman problem: An overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, Jun. 2006.

[14] 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.

[15] F. Wang, Y. Tao, and N. Shi, "A survey on vehicle routing problem with loading constraints," in *Proc. Int. Joint Conf. Comput. Sci. Optim.*, vol. 2, Apr. 2009, pp. 602–606.

[16] S. N. Kumar and R. Panneerselvam, "A survey on the vehicle routing problem and its variants," *Intell. Inf. Manage.*, vol. 4, no. 3, pp. 66–74, 2012.

[17] J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera, and A. A. Juan, "Rich vehicle routing problem: Survey," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 1–28, 2014.

[18] H. Huang, A. V. Savkin, and C. Huang, "Reliable path planning for drone delivery using a stochastic time-dependent public transportation network," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 4941–4950, Aug. 2021.

[19] G. Kim, Y. S. Ong, T. Cheong, and P. S. Tan, "Solving the dynamic vehicle routing problem under traffic congestion," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2367–2380, Aug. 2016.

[20] L. Chen, W.-C. Chiang, R. Russell, J. Chen, and D. Sun, "The probabilistic vehicle routing problem with service guarantees," *Transp. Res. E, Logistics Transp. Rev.*, vol. 111, pp. 149–164, Mar. 2018.

[21] P. Maini, K. Sundar, M. Singh, S. Rathinam, and P. B. Sujit, "Cooperative aerial–ground vehicle route planning with fuel constraints for coverage applications," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 6, pp. 3016–3028, Dec. 2019.

[22] J. Chen, C. Du, X. Lu, and K. Chen, "Multi-region coverage path planning for heterogeneous unmanned aerial vehicles systems," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2019, pp. 356–361.

[23] B. Xin, G.-Q. Gao, Y.-L. Ding, Y.-G. Zhu, and H. Fang, "Distributed multi-robot motion planning for cooperative multi-area coverage," in *Proc. 13th IEEE Int. Conf. Control Autom. (ICCA)*, Jul. 2017, pp. 361–366.

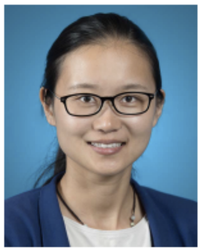
[24] J. Xie, L. R. G. Carrillo, and L. Jin, "An integrated traveling salesman and coverage path planning problem for unmanned aircraft systems," *IEEE Control Syst. Lett.*, vol. 3, no. 1, pp. 67–72, Jan. 2019.

[25] J. Xie, L. Jin, and L. R. G. Carrillo, "Optimal path planning for unmanned aerial systems to cover multiple regions," in *Proc. AIAA Scitech Forum*, Jan. 2019, p. 1794.

[26] J. Xie, L. R. G. Carrillo, and L. Jin, "Path planning for UAV to cover multiple separated convex polygonal regions," *IEEE Access*, vol. 8, pp. 51770–51785, 2020.

[27] 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)*, Jun. 2018, pp. 899–904.

[28] G. Laporte, Y. Nobert, and S. Taillefer, "A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem," *Math. Model.*, vol. 9, no. 12, pp. 857–868, 1987.
[29] S. Almoustafa, S. Hanafi, and N. Mladenović, "New exact method for large asymmetric distance-constrained vehicle routing problem," *Eur. J. Oper. Res.*, vol. 226, no. 3, pp. 386–394, May 2013.
[30] G. Laporte, Y. Nobert, and M. Desrochers, "Optimal routing under capacity and distance restrictions," *Oper. Res.*, vol. 33, no. 5, pp. 1050–1073, Oct. 1985.
[31] C.-L. Li, D. Simchi-Levi, and M. Desrochers, "On the distance constrained vehicle routing problem," *Oper. Res.*, vol. 40, no. 4, pp. 790–799, Aug. 1992.
[32] I. Kara, "Arc based integer programming formulations for the distance constrained vehicle routing problem," in *Proc. 3rd IEEE Int. Symp. Logistics Ind. Informat.*, Aug. 2011, pp. 33–38.
[33] B. L. Golden, T. L. Magnanti, and H. Q. Nguyen, "Implementing vehicle routing algorithms," *Networks*, vol. 7, no. 2, pp. 113–148, 1977.
[34] I. Kara, "On the Miller-Tucker-Zemlin based formulations for the distance constrained vehicle routing problems," in *Proc. AIP Conf.*, vol. 1309, no. 1. College Park, MA, USA: Amer. Inst. Phys., 2010, pp. 551–561.
[35] I. Kara, T. Derya, T. E. Simos, G. Psihoyios, C. Tsitouras, and Z. Anastassi, "Polynomial size formulations for the distance and capacity constrained vehicle routing problem," in *Proc. AIP Conf.*, vol. 1389, no. 1. College Park, MA, USA: Amer. Inst. Phys., 2011, pp. 1713–1718.
[36] G. Laporte, M. Desrochers, and Y. Nobert, "Two exact algorithms for the distance-constrained vehicle routing problem," *Networks*, vol. 14, no. 1, pp. 161–172, 1984.
[37] Z. H. Ahmed, "A lexisearch algorithm for the distance-constrained vehicle routing problem," *J. Math. Comput. Methods*, vol. 1, pp. 1–10, Mar. 2016.
[38] V. Nagarajan and R. Ravi, "Approximation algorithms for distance constrained vehicle routing problems," *Networks*, vol. 59, no. 2, pp. 209–214, Mar. 2012.
[39] C. Bazgan, R. Hassin, and J. Monnot, "Approximation algorithms for some vehicle routing problems," *Discrete Appl. Math.*, vol. 146, no. 1, pp. 27–42, Feb. 2005.
[40] S. Almoustafa, "Distance-constrained vehicle routing problem: Exact and approximate solution (mathematical programming)," Ph.D. dissertation, School Inf. Syst., Comput. Math., Brunel Univ., Uxbridge, U.K., 2013.
[41] K. Sundar, S. Venkatachalam, and S. Rathinam, "Analysis of mixed-integer linear programming formulations for a fuel-constrained multiple vehicle routing problem," *Unmanned Syst.*, vol. 5, no. 4, pp. 197–207, Oct. 2017.
[42] D. Levy, K. Sundar, and S. Rathinam, "Heuristics for routing heterogeneous unmanned vehicles with fuel constraints," *Math. Problems Eng.*, vol. 2014, pp. 1–12, Apr. 2014.
[43] K. Sundar and S. Rathinam, "Route planning algorithms for unmanned aerial vehicles with refueling constraints," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 3266–3271.
[44] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient Boustrophedon multi-robot coverage: An algorithmic approach," *Ann. Math. Artif. Intell.*, vol. 52, nos. 2–4, pp. 109–142, Apr. 2008.
[45] A. Barrientos *et al.*, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *J. Field Robot.*, vol. 28, no. 5, pp. 667–689, 2011.
[46] N. Hazon and G. A. Kaminka, "Redundancy, efficiency and robustness in multi-robot coverage," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 735–741.
[47] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Trans. Robot. Autom.*, vol. 15, no. 5, pp. 918–933, Oct. 1999.
[48] I. A. Wagner, Y. Altshuler, V. Yanovski, and A. M. Bruckstein, "Cooperative cleaners: A study in ant robotics," *Int. J. Robot. Res.*, vol. 27, no. 1, pp. 127–151, Jan. 2008.
[49] H. Choset, "Coverage for robotics—A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, nos. 1–4, pp. 113–126, 2001.
[50] 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.
[51] S. Anbuudayasankar, K. Ganesh, and S. Mohapatra, "Survey of methodologies for *TSP* and *VRP*," in *Models for Practical Routing Problems in Logistics*. Cham, Switzerland: Springer, 2014, pp. 11–42.
[52] M. Gendreau, G. Laporte, and J.-Y. Potvin, "Metaheuristics for the capacitated VRP," in *The Vehicle Routing Problem*. Philadelphia, PA, USA: SIAM, 2002, pp. 129–154.
[53] P. Toth and D. Vigo, *The Vehicle Routing Problem*. Philadelphia, PA, USA: SIAM, 2002.
[54] R. Lima and E. Seminar, "IBM ILOG CPLEX-what is inside of the box," in *Proc. EWO Seminar*, 2010, pp. 1–72.
[55] B. Bixby, "The Gurobi optimizer," *Transp. Re-Search B*, vol. 41, no. 2, pp. 159–178, 2007.
[56] J. D. C. 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, Dec. 1963.
[57] D. L. Miller and J. F. Pekny, "Exact solution of large asymmetric traveling salesman problems," *Science*, vol. 251, no. 4995, pp. 754–761, 1991.
[58] J. Gromicho, J. Paixão, and I. Bronco, "Exact solution of multiple traveling salesman problems," in *Combinatorial Optimization*. Berlin, Germany: Springer, 1992, pp. 291–292.
[59] J. K. Lenstra and A. H. G. R. Kan, "Some simple applications of the travelling salesman problem," *J. Oper. Res. Soc.*, vol. 26, no. 4, pp. 717–733, Dec. 1975.
[60] A. Singh and A. S. Baghel, "A new grouping genetic algorithm approach to the multiple traveling salesperson problem," *Soft Comput.*, vol. 13, no. 1, pp. 95–101, Jan. 2009.
[61] L. Tang, J. Liu, A. Rong, and Z. Yang, "A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan iron & steel complex," *Eur. J. Oper. Res.*, vol. 124, no. 2, pp. 267–282, Jul. 2000.
[62] Y.-B. Park, "A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines," *Int. J. Prod. Econ.*, vol. 73, no. 2, pp. 175–188, 2001.
[63] A. E. Carter and C. T. Ragsdale, "A new approach to solving the multiple traveling salesperson problem using genetic algorithms," *Eur. J. Oper. Res.*, vol. 175, no. 1, pp. 246–257, Nov. 2006.
[64] K. Miettinen, M. M. Mäkelä, and J. Toivanen, "Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms," *J. Global Optim.*, vol. 27, no. 4, pp. 427–446, 2003.
[65] E. Mezura-Montes and C. A. Coello-Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm Evol. Comput.*, vol. 1, no. 4, pp. 173–194, Dec. 2011.
[66] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. White Plains, NY, USA: Longman, 1989.
[67] R. Sivaraj and T. Ravichandran, "A review of selection methods in genetic algorithm," *Int. J. Eng. Sci. Technol.*, vol. 3, no. 5, pp. 3792–3797, 2011.



Junfei Xie (Senior 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 system, and air traffic flow management. She was a recipient of the NSF CAREER Award.

Jun Chen (Member, IEEE) received the B.S. degree in aeronautics engineering from Beihang University, China, and the M.S. and Ph.D. degrees in aerospace engineering from Purdue University.

He is currently an Assistant Professor in aerospace engineering with San Diego State University. His research interests include control and optimization for large-scale networked dynamical systems, with applications in mechanical and aerospace engineering, such as air traffic control, traffic flow management, and autonomous air/ground vehicle systems.