Coverage Path Planning for a UAS Imagery Mission using Column Generation with a Turn Penalty

Younghoon Choi*, Youngjun Choi†, Simon Briceno‡, and Dimitri N. Mavris§

Abstract—**This paper introduces a novel Coverage Path Planning (CCP) algorithm for a Unmanned Aerial Systems (UAS) imagery mission. The proposed CPP algorithm is a vehicle-routing-based approach using a column generation method. In general, one of the main issues of the traditional arc-based vehicle routing approaches is imposing a turn penalty in a cost function because a turning motion of vehicle requires the more amount of energy than a cruise motion. However, the conventional vehicle-routing-based approaches for the CPP cannot capture a turning motion of the vehicle. This limitation of the arc-based mathematical model comes from the property of turning motions, which should be evaluated from two arcs because a turn motion occurs at a junction of the arcs. In this paper, to mitigate the limitation, a route-based model using column generation approach with a turn penalty is proposed. To demonstrate the proposed CPP approach, numerical simulations are conducted with a conventional CPP algorithm.**

I. Introduction

Over the past decade, the market of UAS have been drastically grown in various market areas: military, industry, and hobbyist markets. Due to advanced technologies, the usage of UAS has been broadened from military applications to commercial applications such as agriculture, facility inspection, rescue, and package delivery missions. For those missions, a task of operation planning is crucial to satisfy endurance and range constraints of vehicles. Particularly, an imagery mission needs an optimal coverage path in terms of energy efficiency to maximize the area covered by each vehicle, or in terms of flight time to minimized the mission time. A task that finds an optimal coverage path is called Coverage Path Planning (CPP). The CPP algorithms using a UAS platform have actively researched in various UAS applications such as image mosaicing [1], post-earthquake assessment [2], and 3D terrain reconstruction [3].

In a point of view on the optimization area, the CPP is a similar problem to Chinese postman problem [4]. The Chinese postman problem finds the shortest path that visits every neighborhood (or edge) at least once, whereas the CPP finds an optimal route that passes over all waypoints in an entire Area of Interest (AOI). If a union of the edges in the Chinese postman problem covers all the AOI in the CPP, the two problems become almost identical conceptually. However, the optimal path from Chinese postman problem does not consider any vehicle characteristics such as vehicle speed, and energy. Particularly, for the aerial imagery mission, those vehicle characteristics must be considered to guarantee that the optimal path is feasible in terms of vehicle performances.

In the context of the path optimization, the typical path-planning algorithm can be grouped by five methods: geometric methods, stochastic methods, road map methods, potential field methods, and optimization-based methods [5]. For the imagery CPP problem, the common approach is a grid-based path optimization since the grid can be defined by the camera resolution and UAS operational altitude. The notable grid-based path optimizations can be grouped by exact and approximate methods.

The exact method generally applies back-and-forth motions to cover an entire AOI and finds an optimal line sweep direction introduced by Huang [6] to reduce the number of turns. If the AOI is not convex, the traditional exact solution may not produce a feasible route. To solve this issue, Li et al. [7] introduce an improved exact cellular decomposition method that generates multiple convex sub-areas and applies the exact method in each sub-area. However, this method has a limitation that the mission ending point is automatically determined by the algorithm. In other words, a user cannot select the mission ending point. Torres et al. [3] develop an improved exact method that optimizes the coverage path with the mission starting and ending points selected by the user. For multi-UAVs missions, however, the exact method still needs to solve a high-level optimization problem that determines an optimal sequence visiting all the sub-area.

The representative approximate methods are wavefront-based algorithms, heuristic algorithms, or vehicle-routing-based approaches. Valente et al. [1] and Nam et al. [8] implement the wavefront-based algorithm to obtain the images of an target site using a single UAV. Barrientos et al. [9] adopt the wavefront-based algorithm for a multi-UAV mission to address a large scanning area. In their study, the algorithm decomposes multiple subareas and then solves the wavefront algorithm for each sub-area to obtain the optimal scanning route. Because the two-layer structure, the area decomposition and the wavefront algorithm, has no

*Graduate Researcher, School of Aerospace Engineering, Georgia Institute of Technology, younghoon.choi@gatech.edu

†Research Engineer, School of Aerospace Engineering, Georgia Institute of Technology, ychoi95@gatech.edu

‡Senior Research Engineer, School of Aerospace Engineering, Georgia Institute of Technology, briceno@gatech.edu

§S.P. Langley Distinguished Regents Professor, School of Aerospace Engineering, Georgia Institute of Technology, dimitri.mavris@aerospace.gatech.edu

any interaction between the results of sub-areas, the result of the method could not be better than that of a multi-vehicle optimization problem method that finds a solution without decomposing the whole AOI into multiple small areas. The heuristic algorithms such as genetic algorithms, ant colony optimizations, and particle swarm optimizations have been used to solve the CPP problems [10] since the heuristic algorithm can easily handle discrete and continuous variables. However, these heuristic methods inherently have a limitation that is unclear when the optimization process stops because the heuristic methods cannot check its optimality. Avellar et al. [11] and Nedjati et al. [2] introduce vehicle-routing-based approaches that can solve the CPP problem for multi-vehicle missions. The advantage of the approaches can solve a multi-vehicle CPP problem without any area decompositions, whereas their drawback is that they cannot deal with the number of turns which is directly related to energy consumption of vehicles [6], [10], [12]–[15].

In this paper, we propose a vehicle-routing-based method that finds an optimal coverage path by directly handling the turn motion in the optimization model. The proposed model is based on the mathematical optimization model for the Distance Constrained Vehicle Routing Problem (DVRP) proposed by Kek et al [16] and Kara [17] and modifies it to handle a multi-UAVs problem as well as the number of turns of a vehicle. To solve the model, column generation is utilized with a turn penalty function.

This paper is organized as follows: Section II introduces a column generation method with relevant works. Section III presents the an arc-based optimization model and the route-based optimization model for the DVRP. Section IV proposes the method to solve CPP problems based on the column generation method with turn penalty functions. Section V illustrates two numerical simulations to validate the proposed method. This paper ends with conclusions in Section VI.

II. Related Work: Column Generation

The vehicle-routing-based method is a kind of Traveling Salesman Problem (TSP) that is a decision problem to determine how to visit every city exactly once with the shortest tour distance. The main concept of the vehicle-routing-based method of the CPP is that a union of subareas represented by a waypoint should cover the entire AOI. The difference between the TSP for the CPP and the Chinese postman problem for the CPP is what covers all the AOI, nodes or edges. The network of the VRP is modeled by a graph defined by nodes and edges (or vertices and arcs). The VRP model uses integer variables associated with each arc as design variables, hence, it is an Integer Programming (IP) problem. Column generation is a framework to obtain a solution of an IP problem. Column generation is a main approach to solve huge IP problems in logistics and operational research areas, but has rarely been applied to handle the CPP problem.

Dantzig and Wolfe [18] suggest the fundamental ideas of the column generation. As the first application, Gilmore and Gomory [19], [20] show how column generation can be used to solve a cutting stock problem. Then, Desrochers and Soumis [21] solve an urban transit crew scheduling problem through applying the column generation method. Desrochers et al. [22] suggest a column generation formulation for the Vehicle Routing Problem with Time Windows (VRPTW). Vanderbeck and Wolsey [23] present a column generation approach for general IP problems with integer variables, not binary variables that allow only 0 or 1, by combining branch and bound and column generation. In UAS applications, Mufalli et al. [24] and Zillies et al. [25] show how the column generation can be utilized to solve the vehicle routing problems, but solve only vehicle routing problems associated with surveillance missions that are not a kind of the coverage path problem.

A network of the VRP is described by vertices and arcs (or nodes and edges) defined in the graph theory. A vertex represents a location such as a city and a waypoint, and an arc describes a movement of a vehicle from a vertex to another vertex. In a myriad of the VRPs, an arc has been utilized as a design variable. However, the VRP model used in column generation needs to be formulated by design variables that describe a route or path, a set of arcs. Column generation splits the standard VRP problem into the master problem and the sub-problem using Dantzig and Wolfe decomposition [22].

The master problem directly handles a route as a design variable. The route-based model has a weakness that the number of routes exponentially increases as the number of node increases. Hence, the computation resource that depends on the problem size could be an obstacle to solving the master problem. To mitigate this weakness, the master problem deals with a subset of all routes during column generation process, which is called the restricted master problem. The master problem can be modeled by a set partitioning problem or a set covering problem which is a relaxation of a set partitioning problem [18], [26].

The structure of the sub-problem depends on that of the master problem. First, the master problem modeled as the set partitioning problem works with the Elementary Shortest Path Problem with Resource Constraints (ESPPRC) as a sub-problem. The ESPPRC only deal with elementary routes, which allow a route to visit each node exactly once. Second, the mater problem designed as the set covering problem runs with the Shortest Path Problem with Resource Constraints (SPPRC) as a sub-problem. This SPPRC allows a route to visit a node more than once, which is a relaxation of the ESPPRC. The result of the sub-problem is utilized to create candidate routes added to the master problem. To circumvent the overlapped coverage routes, this paper adopts the set partitioning problem as a master problem, and the ESPPRC as a sub-problem.

III. Mathematical Optimization Model

For the CPP optimization formulation, we extend the Kara's DVRP model [17]. To be more specific, the Kara's DVRP model optimizes flight range with the fixed number of vehicles, but our extended DVRP model can minimize the number of vehicles, and the total flight range of the vehicles as well. The extended DVRP model is converted to a route-based optimization model that allows us to include a turning penalty, which cannot be solve by the arc-based Kara's DVRP model. To solve the route-based DVRP model, we applies a column generation technique. The subsections introduce the extended arc-based DVRP formulation, and the route-based DVRP formulation to use the column generation technique.

A. Arc-based Optimization Model

The DVRP is modeled based on a graph, G, and a fleet of UAVs, V. The graph consists of a set of nodes, N, a set of arcs, A. The set of nodes $N = \{0, 1, 2, \cdots, n+1\}$ includes the starting depot, 0, the returning depot, $n+1$, and waypoints, $W = \{1, 2, \cdots, n\}$. In the UAS imagery mission, the waypoints can be defined by sensor scanning locations. The set of arcs, $A = \{(i,j) : i,j \in N, i \neq j\}$, represents the connection between two nodes. The cost c_{ijk} of each arc corresponds flight distance from the node i to the node j by the vehicle k, d_{ijk}. The extended model uses a set of design variables, x_{ijk}. If the vehicle k travels along the arc (i,j), x_{ijk} is defined as 1, otherwise 0. To capture the number of vehicles, three-index formulations are adopted for the extended DVRP model. The arc-based optimization model can be written by

$$\text{Minimize} \quad \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ijk} x_{ijk} \tag{1}$$

Subject to

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1 \qquad (\forall i \in W) \tag{2}$$

$$\sum_{j \in N} x_{0jk} = 1 \qquad (\forall k \in V) \tag{3}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0 \qquad (\forall h \in W,\ \forall k \in V) \tag{4}$$

$$\sum_{i \in N} x_{i(n+1)k} = 1 \qquad (\forall k \in V) \tag{5}$$

$$\sum_{j \in N} s_{ijk} - \sum_{j \in N} s_{jik} - \sum_{j \in N} d_{ijk} x_{ijk} = 0 \qquad (\forall i \in N,\ \forall k \in V) \tag{6}$$

$$y_{0jk} = d_{0jk} x_{0jk} \qquad (\forall j \in N,\ \forall k \in V) \tag{7}$$

$$y_{ijk} \leq (D - d_{j0k}) x_{ijk} \qquad (\forall j \in N,\ \forall k \in V) \tag{8}$$

$$y_{i(n+1)k} \leq D x_{i(n+1)k} \qquad (\forall i \in N,\ \forall k \in V) \tag{9}$$

$$y_{ijk} \geq (d_{0ik} + d_{ijk}) x_{ijk} \qquad (\forall i,j \in N,\ \forall k \in V), \tag{10}$$

where s_{ijk} is a flow variable that measures the total distance traveled by the vehicle k from the starting depot to the node j when it goes from i to j. The objective function of the model, Eq. (1), is defined as minimizing the total cost that is proportional to the flight distance or time traveled by vehicles. The following operating conditions are defined as constraints: All the waypoint should be visited exactly once, Eq. (2). Each vehicle should start at the starting depot, Eq. (3), and after visiting a waypoint, it has to leave the waypoint, Eq. (4), and it should return to the returning depot, Eq. (5). All sub-tours should be removed by constraint Eq. (6). Each vehicle is allowed to travel up to the maximum distance described by D, which is guaranteed by constraints Eq. (7) - (10).

B. Route-based Optimization Model

The constraint matrix of the proposed arc-based optimization model has a block angular structure of which some constraints are related only specific design variables. By the Dantzig-Wolfe decomposition [18], the block-angular-structure optimization model can be decomposed into two problems: the first one includes just coupled constraints and the second one has non-coupled constraints only. Note that while the constraint of the arc-based optimization model, Eq. (2), is coupled with the vehicles, the others are independent from the vehicles. This implies the model has block angular structure. Using this property, the model can be decomposed into the master problem and the sub-problem. Moreover, the master problem can be reformulated to a route-based optimization model for the column generation method according to Minkowski-Weyl theorem [27].

1) Master Problem: To obtain the route-based optimization model, a new design variable need to be defined for each route, not arc. Let P^k be a set of feasible routes of vehicle k, $k \in V$. A variable c_p^k is defined as a cost of the route p traveled by the vehicle k, $p \in P^k,\ k \in V$. A variable a_{ip}^k is a visiting status of the route, p, of the vehicle, k. Let a_{ip}^k be 1 if the vehicle k visits the waypoint i when traveling along route p, and a_{ip}^k be 0, otherwise. A route-based design variable, y_p^k, is defined as 1 if the route p is chosen by vehicle k, $p \in P^k,\ k \in V$, or 0, otherwise. Because y_p^k is a 0-1 integer variable, the master problem is an IP problem that can be written by

$$\text{Minimize} \quad \sum_{k \in V} \sum_{p \in P^k} c_p^k y_p^k \tag{11}$$

Subject to

$$\sum_{k \in V} \sum_{p \in P^k} a_{ip}^k y_p^k = 1 \qquad (\forall i \in W) \tag{12}$$

$$\sum_{p \in P^k} y_p^k = 1 \qquad (\forall k \in V) \tag{13}$$

$$y_p^k \in \{0, 1\} \qquad (\forall k \in V,\ \forall p \in P^k). \tag{14}$$

To solve an IP problem, a linear relaxation needs to be applied. By converting an IP problem to a Linear Programming (LP) problem, a myriad of methods based on a simplex algorithm can be utilized to solve it. Column generation, also, has the benefit because it needs to use a LP model instead of an IP model as the master problem. By a linear relaxation, y_p^k is converted from a 0-1 integer variable to a continuous design variable. The LP model of the master problem is defined as follows

$$\text{Minimize} \qquad \sum_{k\in V}\sum_{p\in P^k} c_p^k y_p^k \tag{15}$$

Subject to

$$\sum_{k\in V}\sum_{p\in P^k} a_{ip}^k y_p^k = 1 \qquad (\forall i \in W) \tag{16}$$
$$\sum_{p\in P^k} y_p^k = 1 \qquad (\forall k \in V) \tag{17}$$
$$y_p^k \geq 0 \qquad (\forall k \in V,\ \forall p \in P^k). \tag{18}$$

This master problem is a generalized formulation that can solve multi-depot or heterogeneous fleet problems. For simplicity, let us consider a single depot and homogeneous fleet problem. Then, the master problem can be simplified as follows

$$\text{Minimize} \qquad \sum_{p\in P'} c_p y_p \tag{19}$$

Subject to

$$\sum_{p\in P'} a_{ip} y_p = 1 \qquad (\forall i \in W) \tag{20}$$
$$y_p \geq 0 \qquad (\forall p \in P^{'}), \tag{21}$$

where, $P^{'} \subset P$, is a subset of feasible solutions, which is candidate solutions in the column generation. Thus, This model for the master problem is the restricted master problem that handles a subset of routes, not all possible routes. For convenience, this paper just calls it the master problem.

2) Sub-problem: The sub-problem consists the objective function and the constraints that are dependent on vehicles. For a heterogeneous fleet problem, the sub-problems need to be as many as the number of vehicle types. Moreover, each sub-problem should be solved for every iteration of column generation process. For a homogeneous fleet problem, however, a single sub-problem needs to be solved in an iteration of column generation process. Note that when solving the sub-problem, the modified cost, $\hat{c}_{ij}$ is used for each arc, (i,j), where $\hat{c}_{ij} = c_{ij} - \pi_i$, π_i is a dual variable of the master problem with $i \in W$. The ESPPRC with distance constraints, the sub-problem, is modeled as follows

$$\text{Minimize} \qquad \sum_{i\in N}\sum_{j\in N} \hat{c}_{ij} x_{ij} \tag{22}$$

Subject to

$$\sum_{j\in N} x_{0j} = 1 \tag{23}$$
$$\sum_{i\in N} x_{ih} - \sum_{j\in N} x_{hj} = 0 \qquad (\forall h \in W) \tag{24}$$
$$\sum_{i\in N} x_{i(n+1)} = 1 \tag{25}$$
$$\sum_{j\in N} s_{ij} - \sum_{j\in N} s_{ji} - \sum_{j\in N} d_{ij} x_{ij} = 0 \qquad (\forall i \in N) \tag{26}$$
$$y_{0j} = d_{0j} x_{0j} \qquad (\forall j \in N) \tag{27}$$
$$y_{ij} \leq (D - d_{j0}) x_{ij} \qquad (\forall j \in N) \tag{28}$$
$$y_{i(n+1)} \leq D x_{i(n+1)} \qquad (\forall i \in N) \tag{29}$$
$$y_{ij} \geq (d_{0i} + d_{ij}) x_{ij} \qquad (\forall i, j \in N). \tag{30}$$

Constraints Eq. (23) - (25) are flow constraints for a route from the starting depot to the returning depot. Constraint Eq. (26) removes sub-tours. The others guarantee to satisfy travel distance conditions.

IV. Column Generation with a Turn Penalty for Vehicle Routing Problems

The column generation solves integer problems such as the cutting stock and a vehicle routing problem. In general, the column generation for VRP consists of three parts: an initial value problem, a master problem, and a sub-problem. The initial value problem obtains feasible routes that is used to create the initial candidate routes for the master problem. The master problem determines an optimal route for each vehicle from candidate routes. The subproblem generates a set of new candidate routes. These candidate routes are added in the master problem. The master problem with the updated candidate routes is solved in the next iteration. Note that unlike the arc-based vehicle routing problem, the column generation can count the number of turns of all the candidate routes. This is because a route of the arc-based model is determined after optimization process, whereas a route of the route-based model is determined during optimization process. Therefore, the column generation approach can optimize the number of turns and distance as well.

A. Initial Value Problem

The initial value problem identifies a set of feasible routes in the first iteration. The identified feasible routes become design variables in the master problem. In general, the initial value problem can be solved by two methods: trivial solution method and savings algorithm [28]. The trivial solution method simply creates routes through connecting three nodes:the starting depot, a waypoint, and the returning depot. Consequently, the number of the routes are same as the number of the waypoints. On the other hand, the savings algorithm, a simple heuristic approach, generates a set of greedy routes. This algorithm produces less number of routes than the trivial solution. Note that the number of initial feasible routes can be used as the number of initial vehicles. Thus, the savings algorithm can improve computational efficiency of column generation through reducing the number of initial feasible routes. Because of this benefit, this paper selects the savings algorithm to solve the initial value problem.

B. Master Problem with A Turn Penalty

After solving the initial value problem, the master problem solves the LP problem in the column generation process, which is the linear relaxation of the IP problem, Eq. (19) - Eq. (21). The LP problem computes dual variables that guides the sub-problem to specify better routes compared to the existing candidate routes. The new routes could reduce the value of the objective function in the master problem. In the objective function, we revise the cost function to capture a turn penalty as follows

$$c_p^{'} = \sum_{(i,j)\in A} c_{ij}x_{ij} + f_{penalty}(p) \quad (\forall p \in P^{'}), \tag{31}$$

where $c_p^{'}$ is a new cost function, and $f_{penalty}(p)$ is a penalty term that has a non-linear function with respect to the route p. To consider the number of turns, the penalty function $f_{penalty}(p)$ is defined by

$$f_{penalty}(p) \ = \ T_p * c_{penalty} \qquad (\forall p \in P^{'}), \tag{32}$$

where Tp represents the number of turns on route p, and $c_{penalty}$ is a turning cost.

C. Sub-problem with Turn Penalty

The goal of solving the sub-problem is to identify candidate routes that potentially reduce the value of the objective function in the master problem. The traditional method solving the sub-problem is a label correcting algorithm suggested by Desrochers et al. [22], which a kind of dynamic programming techniques. To solve the ESPPRC, Feillet et al. [29] proposed the modified label correcting algorithm through introducing the concept of a unreachable node. The unreachable node is a node that is already visited on a route or the vehicle cannot reach the node because of the limited resource of the vehicle.

In the proposed label correcting algorithm, the vehicle resource is defined as $R_z = (D_z, s_z, U_z^0, \cdots, U_z^{n+1})$, where D_z is flight distance traveled by the path from the starting depot to the node z, s_z is the number of unreachable nodes, and $U_z^0, \cdots, U_z^{n+1}$ is the vector of unreachable nodes. Then, the label is defined as $L_z = (R_z, Q_z, C_z)$, where Q_z indicates the path from the starting depot to the node z, and the term C_z is the cumulative cost from the starting depot to the node z along the Q_z, which is defined by

$$C_z = \sum_{(i,j)\in Q_z} c_{ij} + f_{penalty}(Q_z). \tag{33}$$

The path information Q_z is just used to count the number of turns in the penalty function.

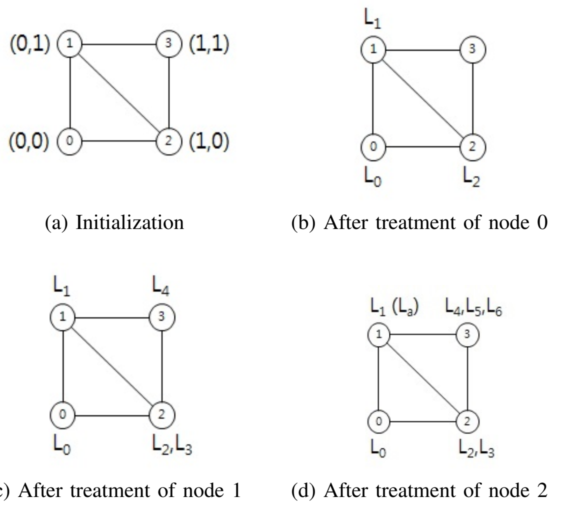
The proposed label correcting algorithm uses labels that include the information about resources, path and cost. Each label records the accumulated information from the starting depot to the returning depot depending on only its previous label using a dynamic programming approach. Using the accumulated label information, the algorithm identifies a set of the feasible routes resulting from the labels on the returning depot.

One of the main concepts of the proposed label correcting algorithm is dealing with only elementary paths, which visit a node exactly once. In order to implement this concept, the vector of unreachable nodes, $U_z^0, \cdots, U_z^{n+1}$, is defined on a new label. The new label is not pinned at a node when the node considered to visit is already visited, or the resource of vehicle is insufficient to reach the node due to maximum endurance or range. This concept guarantees that each label has only an elementary path.

Another main concept is a process to reduce the number of labels through finding non-dominated labels suggested by Feillet et al. [29]. The non-dominated labels are specified by the concept of Pareto frontier is utilized, which is a common method in multi-objective optimization. To be more specific, let us consider two labels, $L_x = (D_x, s_x, U_x^0, \cdots, U_x^{n+1}, Q_x, C_x)$ and $L_y = (D_y, s_y, U_y^0, \cdots, U_y^{n+1}, Q_y, C_y)$. If $D_x \le D_y$, $s_x \le s_y$, $U_x^0 \le U_y^0$, $\cdots$, $U_x^{n+1} \le U_y^{n+1}$, and $C_x \le C_y$, then L_x dominates L_y. As a result, the label L_x is a non-dominated label. In contrast, if one of the conditions is not satisfied, L_x and L_y do not dominate each other. Hence, L_x and L_y are considered as non-dominated labels. The process fining non-dominant labels is applied at each node. The results of all the non-dominated labels are continuously stored until the labels on the returning depot is completed, which lead to generates feasible routes.

The label correcting algorithm starts with an initial label. The nodes that need to be visited are stored in a set queue E, which is a data structure with the concept of First In First Out (FIFO) ensuring the uniqueness of its elements. The set queue E determines a starting node during the iteration. The algorithm checks potential new labels using the labels on the starting node and the list of its neighbors. The new labels are created when the node can move from the starting node to its neighbor nodes. In other words, the neighbor node is not an unreachable node and the constraint, which is a vehicle resource, is not violated. Then, the algorithm only collects non-dominant labels from the new labels. The algorithm is finished when there is no additional route discovered. Algorithm 1 is the pseudo code of the label correction algorithm, where $label_i$ is a label at node i, $list\_of\_labes$ is all labels at node i, $enqueue()$ is a function to insert a node into the set queue E, $dequeue()$ is a function to obtain a node from the set queue E, $extend()$ is a function to find a new label based on $label_i$ at v_j, and $EFF()$ is a function to search a set of non-dominated labels.

Let us consider a simple example of the label correcting



(a) Initialization (b) After treatment of node 0

(c) After treatment of node 1 (d) After treatment of node 2

Fig. 1. Illustration of the first iteration of label correcting algorithm.

TABLE I
LABEL LIST OF A SIMPLE EXAMPLE

	Data							
Label	D_z	s_z	U_z^0	U_z^1	U_z^2	U_z^3	Q_z	C_z
L_0	0	1	1	0	0	0	[0]	0
L_1	1	2	1	1	0	0	[0,1]	1
L_2	1	2	1	0	1	0	[0,2]	1
L_3	2.414	3	1	1	1	0	[0,1,2]	3.414
L_4	2	3	1	1	0	1	[0,1,3]	3
L_5	2	3	1	0	1	1	[0,2,3]	3
L_6	3.414	4	1	1	1	1	[0,1,2,3]	5.414

algorithm for DVRP depicted in Fig. 1. The example case has an undirected graph with 4 nodes, $N = \{0, 1, 2, 3\}$. The location of each node is $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$. The node 0 is assumed to be the starting depot, and the node 3 is assumed to be the returning depot. We also assume that the graph has 5 arcs, $A = \{(0, 1), (0, 2), (1, 2), (1, 3), (2, 3)\}$. In the cost function, the penalty formulation is assumed by in Eq. (32), and the penalty cost $c_{penalty}$, be 1. The initial label, $L_0 = (0, 1, 1, 0, 0, 0, [0], 0)$, is pinned at the starting depot, and the set queue E has 0. As a result, the first sub-iteration starts at node 0. Because of two neighbors (node 1 and node 2) of the node 0. the two labels are created at each node since L_0, L_1, and L_2 are non-dominated labels. Therefore, the set queue E has two elements, 1 and 2. The second sub-iteration starts at node 1, and check its neighbors (node 0, node 2 and node 3). The node 2 and node 3 are identified as new labels (L_3 and L_4) because the node 0 is a unreachable node. In the first sub-iteration, the set queue E includes two elements (node 2 and node 3). The third sub-iteration starts node 2 with L_2 and L_3. Because of three neighbors (node 0, node 1, and node 3), the algorithm checks feasibility of the all the routes based on an unreachable node and non-dominant route condition. A potential label based on L_2 ($L_a = (2.414, 3, 1, 1, 1, 0, [0, 2, 1], 3.414)$) is rejected because this label is dominated by L_1. Other neighbors are easily checked based on the label checking algorithm illustrated in Algorithm 1. The results of the label correction in the example is summarized on Table I.

Algorithm 1 Pseudo code of label correcting algorithm adopted from [29]

```
Input: $G(N, A)$
Output: $list\_of\_labels_{n+1}$
  Initialization
  $E.enqueue(0)$
  while $E$ is not empty do
    $v_i = E.dequeue()$
    for $v_j$ in neighbors of $v_i$ do
      $F_{ij}$ = empty
      for $label_i$ in $list\_of\_labels_i$ do
        if $U_i^j$ is reachable then
          $F_{ij}.extend(label_i, v_j)$
        end if
      end for
      $list\_of\_labels_j$ = $EFF(F_{ij} \cup list\_of\_labels_j)$
      if $list\_of\_labels_j$ has changed then
        $E.enqueue(v_j)$
      end if
    end for
  end while
  return $list\_of\_labels_{n+1}$
```

To combine the label correcting algorithm into the column generation, the cumulative cost, C_z, needs to be updated by dual variables from the master problem, π. The updating rule of the cumulative cost is defined as follows

$$\hat{C}_z = \sum_{(i,j)\in Q_z} (c_{ij} - \pi_i) + f_{penalty}(Q_z). \tag{34}$$

The updated cumulative cost, $\hat{C}_z$, varies every sub-problem iteration. Thus, the sub-problem finds a different set of route even though any information of the graph is unchanged. If $\hat{C}_z < 0$ at $z = n + 1$, the route in the label is added in the master problem. This concept of the reduced cost commonly applies to find a new basis in simplex algorithms.

D. Column Generation Framework with a Turn Penalty

The column generation consists of the initial value problem, the master problem, and the sub-problem. The details of the column generation is illustrated in Fig. 2. During each iteration of column generation, the linear relaxation of the master problem and the sub-problem are solved.

Let us consider what happens to the mathematical structure of the master problem during column generation process. The constraints of the master problem can be arranged by design variables which is each route. At the i-th iteration of column generation, suppose that the master problem has p design variables. After updating the master problem for $(i + 1)$-th

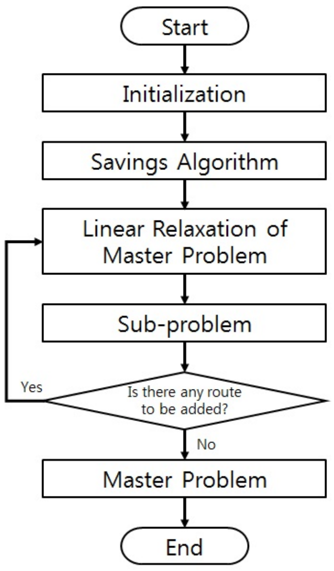


Fig. 2. Process of the proposed method based on column generation.

$$\begin{array}{c|ccccc|}
 & y_1 & y_2 & y_3 & \cdots & y_p \\
\hline
w_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1p} \\
w_2 & a_{21} & a_{22} & a_{23} & \cdots & a_{2p} \\
w_3 & a_{31} & a_{32} & a_{33} & \cdots & a_{3p} \\
\vdots & \vdots & \vdots & \vdots & & \vdots \\
w_n & a_{n1} & a_{n2} & a_{n3} & \cdots & a_{np} \\
\hline
\end{array}$$

(a) i-th iteration

$$\begin{array}{c|cccccccc|}
 & y_1 & y_2 & y_3 & \cdots & y_p & y_{p+1} & \cdots & y_{p'} \\
\hline
w_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1p} & a_{1(p+1)} & \cdots & a_{1p'} \\
w_2 & a_{21} & a_{22} & a_{23} & \cdots & a_{2p} & a_{2(p+1)} & \cdots & a_{2p'} \\
w_3 & a_{31} & a_{32} & a_{33} & \cdots & a_{3p} & a_{3(p+1)} & \cdots & a_{3p'} \\
\vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
w_n & a_{n1} & a_{n2} & a_{n3} & \cdots & a_{np} & a_{n(p+1)} & \cdots & a_{np'} \\
\hline
\end{array}$$

(b) (i+1)-th iteration

Fig. 3. Shapes of constraint matrix at i-th and (i+1)-th iterations of column generation.

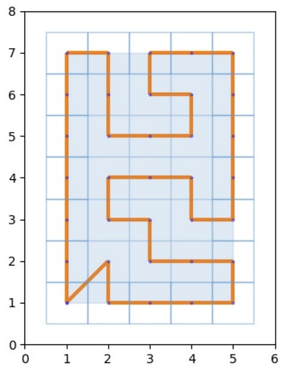


Fig. 4. Solution of the arc-based model for a rectangle problem.

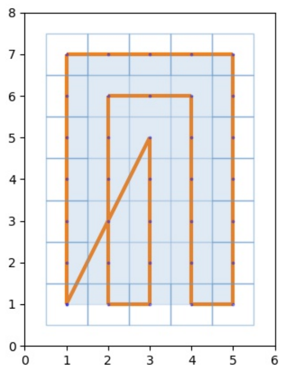


Fig. 5. Solution of the route-based model with a turn penalty for a rectangle problem.

iteration, it has p', $p' > p$, design variables which are added by the sub-problem of i-th iteration as described in Fig. 3. Each design variable, a feasible route, is added from the sub-problem that creates a new column in terms of the constraint matrix, and the information of the route is reflected by the coefficients of the column, a. If there is no design variable added from the sub-problem, $p = p'$, the column generation process is stopped. Then, the proposed framework solves the IP problem to obtain the optimal route.

V. Numerical Simulation

To validate the proposed method, we conduct an experiment with two numerical simulations. For simplicity, it is assumed that the number of UAVs is one. To solve the arc-based model and both the linear relaxation of the mater problem and the IP master problem in the route-baed model, the simulation uses the Gurobi solver that is a commercial optimization solver.

The AOI of the first scenario is a rectangular area that is modeled by 36 nodes: 34 waypoints and 2 depots. One depot is for the starting node and the other depot is for the returning node, which implies one phisical depot. The size of each grid cell is one-by-one. The location of the starting depot and the returning depot are $(1, 1)$. Fig. 4 and Fig. 5 present the results of the arc-based optimization and the route-based optimization. In the figures, the area with light blue color is the AOI, and the line with orange color is the optimized route. Results show that the route of the arc-based optimization model has 35.41 distance and 19 turns. In contrast, the route of the proposed route-based optimization model has 38.47 travel distance and 9 turns. The results are reasonably expected as the cost function of the arc-based optimization model only minimizes distance, but the cost function of the proposed route-based optimization model minimizes distance, and the number of turns. Therefore, the proposed method has less turns, but slightly longer travel distance.

The AOI of the second scenario is applied to the irregular shape that Li et al. [7] uesd for the demonstration of their CPP algorithm. This model has 34 nodes:32 waypoints and 2 depots. The location of the starting/returning depot is $(4, 0)$. Fig. 6 and 7 illustrates the results of both optimization models. The route of the arc-based optimization has 34.24 travel

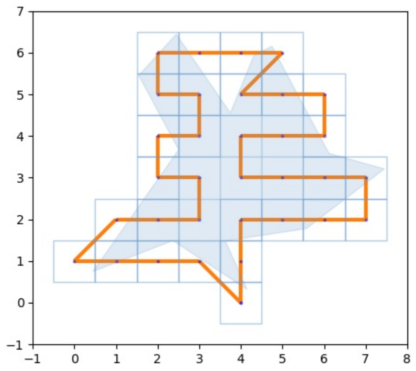


Fig. 6. Solution of the arc-based model for Li's problem.

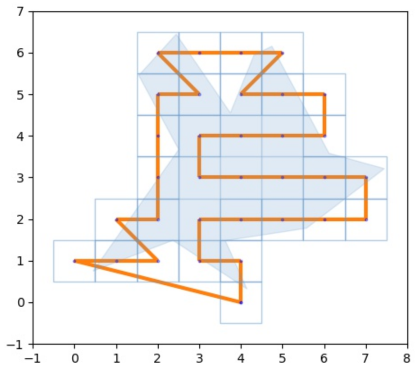


Fig. 7. Solution of the route-based model with a turn penalty for Li's problem.

distance with 20 turns. On the other hand, the proposed route-based optimization has 37.37 travel distance and 18 turns. As expected, the propsoed route-based optimization requires less turns, but longer travel distance because of the penalty term in the cost function. We can also observe that unlike the result of the previous retangular AOI, the difference of the turns between both methods are small because of the irregular shape. It implies that this proposed route-based optimization method can generate more energy effective route when AOI is not irregular shape. Both numerical studies clearly show that the proposed route-based optimization method is effective to reduce the number of turns. All the results of two example studies are summarized in Table II.

VI. Conclusion

The standard arc-based vehicle routing problem for the CPP problem may not generate an energy efficient route because it cannot handle directly turn motions that is one of the main factors associated with energy consumption. To address this standard arc-based vehicle routing problem, this paper proposes a vehicle-routing-based optimization model

TABLE II
Results of the numerical simulations

Geometry	Model	Penalty	Distance	Turns	Route
Rectangle	Arc-based	-	35.41	19	Fig. 4
Rectangle	Route-based	Eq. (32)	38.47	9	Fig. 5
Li's shape	Arc-based	-	34.24	20	Fig. 6
Li's shape	Route-based	Eq. (32)	37.37	18	Fig. 7

that enables us to include a turn penalty in a cost function. The proposed routing-based optimization is solved by the framework of the column generation that consists of the master and sub problems. In the iteration process, the linear relaxation of the master problem is solved, which allows the sub-problem to identify better candidate routes. After finishing the iteration process, the column generation framework solves IP problem to compute the optimal route. Numerical simulations with the arc-based and the proposed route-based optimization models were conducted to compare their performance in terms of travel distance and the number of turns. Results indicate that the proposed route-based optimization method can reduce the number of turns using the penalty term in the cost function. Particularly, when the AOI is not irregular shape, the proposed route-based optimization method is more effective to generate the optimal route with less turns. The other advantage of the proposed method is that it has a flexible penalty function that allows us to easily impose other penalties such as turning angle, turning time, and actual energy consumption.

References

[1] J. Valente, D. Sanz, J. D. Cerro, A. Barrientos, and M. A. de Frutos, "Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields," *Precision Agric*, vol. 14, pp. 115–132, 2013.

[2] A. Nedjati, G. Izbirak, B. Vizvari, and J. Arkat, "Complete coverage path planning for a multi-UAV response system in post-earthquake assessment," *Robotics*, vol. 26, no. 5, 2016.

[3] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres, "Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction," *Expert Systems With Applications*, vol. 55, pp. 441–451, 2016.

[4] J. Edmonds and E. L. Johnson, "Matchig, euler tours and the chinese postman," *Mathematical Programming*, vol. 5, pp. 88–124, 1973.

[5] Y. Choi, H. Jimenez, and D. N. Mavris, "Two-layer obstacle collision avoidance with machine learning for more energy-efficient unmanned aircraft trajectories," *Robotics and Autonomous Systems*, vol. 98, pp. 158–173, 2017.

[6] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, Seoul, Korea, May 21-26 2001.

[7] Y. Li, H. Chen, M. J. Er, and X. Wang, "Coverage path planning for UAVs based on enhanced exact cellular decomposition method," *Mechatronics*, vol. 21, pp. 876–885, 2011.

[8] L. H. Nam, L. Huang, X. J. Li, and J. F. Xu, "An approach for coverage path planning for UAVs," in *Advanced Motion Control (AMC), 2016 IEEE 14th International Workshop*, 22-24 April 2016.

[9] A. Barrientos, J. Colorado, J. del Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: a practical approach to area coverage and path planning for fleets of mini aerial robots," *Journl of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[10] A. Khan, I. Noreen, and Z. Habib, "On complete coverage path planning algorithms for non-holonomic mobile robots: survey and challenges," *Journal of Information Science and Engineering*, vol. 33, pp. 101–121, 2017.

[11] G. S. C. Avellar, G. A. S. Pereira, L. C. A. Pimenta, and P. Iscold, "Multi-UAV routing for area coverage and remote sensing with minimum time," *Sensors*, vol. 15, pp. 27 783–27 803, 2015.

[12] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Arificial Intelligence*, vol. 31, pp. 113–126, 2001.

[13] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, pp. 1258–1276, 2013.

[14] C. D. Franco and G. Buttazzo, "Energy-aware coverage path planning of UAVs," in *Proc. of the IEEE Int. Conf. on Autonomous Robot Systems and Competitions (ICARSC2015)*, Vila Real, Portugal, April 8-10 2015.

[15] Y. Choi, Y. Choi, S. Briceno, and D. N. Mavris, "Three-dimensional uas trajectory optimization for remote sensing in an irregular terrain environment," in *The 2018 International Conference on Unmanned Aircraft Systems*, Dallas, TX, June 12-15 2018.

[16] A. G. H. Kek, R. L. Cheu, and Q. Meng, "Distance-constrained capacitated vehicle routing problem with flexible assignment of start and end depots," *Mathematical and Computer Modelling*, vol. 47, pp. 140–152, 2008.

[17] I. KARA, "Arc based integer programming formulations for distance constrained vehicle routing problem," in *LINDI 2011 - 3rd IEEE International Symposium on Logistics and Industrial Informatics*, Budapest, Hungary, August 25-27 2011.

[18] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programmings," *Operaions Research*, vol. 8, pp. 101–111, 1960.

[19] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations Research*, vol. 9, no. 6, pp. 849–859, 1961.

[20] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem-part 2," *Operations Research*, vol. 11, no. 6, pp. 863–888, 1963.

[21] M. Desrochers and F. Soumis, "A column generation approach to the urban transit crew scheduling problem," *Transportation Science*, vol. 23, no. 1, pp. 1–13, 1989.

[22] M. Desrochers, J. Desposiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, no. 2, pp. 342–354, 1992.

[23] F. Vanderbeck and L. A. Wolsey, "An exact algorithm for IP column generation," *operations Research Letters*, vol. 19, pp. 151–159, 1996.

[24] F. Mufalli, R. Batta, and R. Nagi, "Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans," *Computer & Operations Research*, vol. 39, pp. 2787–2799, 2012.

[25] J. Zillies, S. Westphal, D. Thakur, V. Kumar, G. Pappas, and D. Scheidt, "A column generation approach for optimized routing and coordination of a UAV fleet," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, EPFL, Lausanne, Switzerland, October 23-27 2016.

[26] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, "Branch-and-price: column generationn for solving huge integer probrams," *INFORMS*, vol. 46, no. 3, pp. 316–329, 1998.

[27] M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, Eds., *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer, 2010, ISBN: 978-3-540-68279-0.

[28] G.Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points." *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964.

[29] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, "An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problem," *Networks*, vol. 44, no. 3, pp. 216–229, 2004.