IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 16, 2020, accepted February 19, 2020, date of publication February 24, 2020, date of current version March 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2976122

Node and Edge Drone Surveillance Problem With Consideration of Required Observation Quality and Battery Replacement

IVAN KRISTIANTO SINGGIH, (Member, IEEE), JONGHWA LEE, AND BYUNG-IN KIM
Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), Pohang 37673, South Korea

Corresponding author: Byung-In Kim (bkim@postech.ac.kr)

This work was supported in part by the Brain Korea 21 Postdoctoral Program, in part by the National Research Foundation of Korea, grant funded by the Government of the Republic of Korea under Grant NRF-2019R1F1A1058165, and in part by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE), The Competency Development Program for Industry Specialist, under Grant P0008691.

ABSTRACT Our study introduces a drone routing problem in which drones fly to capture photos for surveillance purposes after a disaster. The drones perform observations on nodes and edges representing populated areas and road segments of a network from multiple altitudes. Each target node and edge requires observation at least once with a certain required quality. When the drones fly at a relatively high altitude, they can simultaneously capture low-quality photos and a large number of observed target nodes and edges. However, high-quality photos and narrow observation areas can be captured from a relatively low altitude. Each drone has a limited battery capacity and thus must return to the depot for battery replacement. This study routes the drones to satisfy the required photo quality of all target nodes and edges while minimizing the makespan of the surveillance by all drones. Our study is the first to examine a multiple-drone routing problem while considering flight altitude-dependent observation quality, battery replacement, node and edge combination, and minimizing the makespan. Our problem is formulated as a mixed integer linear programming (MILP) model. Firefly and adaptive–reactive tabu search algorithms are proposed. The latter outperforms the former and obtains better solutions than those in the MILP model for small-sized instances within a given short computation time.

INDEX TERMS Adaptive–reactive tabu search, altitude, battery replacement, drone routing, firefly algorithm, photo quality, surveillance.

I. INTRODUCTION

As a type of unmanned aerial vehicles (UAVs), drones are used for various purposes, such as parcel delivery, surveillance, and entertainment purposes. Many studies have been conducted on technologies that support the drone operations, such as wireless technologies for communications within drones or between drones and the depot [1], [2], various energy sources [3], [4], vision algorithms [5], cooperative routing between ground vehicles and drones [6], and human–drone interaction system [7].

In this study, a multiple drone point-and-arc-routing problem for post-disaster surveillance is introduced. Each target node and edge requires observation by taking photos with a certain quality (e.g., low, medium, and high) using the drones. The drones can conduct these observations from points or arcs at any multiple altitudes [8], [9]. When the drones fly at a relatively low altitude, they can capture high-quality photos but can only perform observations within a small range. Conversely, flying the drones at a high altitude can only capture low-quality photos but enables the drones to observe more target nodes and edges simultaneously. The surveillance time can be minimized by performing the observations at appropriate altitudes, such as relatively low altitudes to reduce the required travel times and high altitudes to observe more target nodes and edges simultaneously and reduce the number of points requiring visits.

The limited drone battery must be considered during observations. When the battery level is critical, a drone must return to the depot to replace its battery before departing for its

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

next travel. Generating drone routes that allow observing more target nodes and edges is necessary for an effective observation within a short time.

Our study is the first to examine the drone routing problem at more than one altitude for node and edge observation, considering the required observation quality and battery replacement. In our new routing problem, different drone flight altitudes determine the captured photo quality. Our problem is different from a well-known vehicle routing problem (VRP) and arc routing problem, in which vehicles must directly travel to any target node or edge. In our problem, multiple observations of target nodes and edges are possible from a single point visited by a drone.

Our problem is formulated as a mixed integer linear programming (MILP) model. Firefly and adaptive–reactive tabu search (ARTS) algorithms are developed to solve the problem. The initial version of the firefly algorithm was introduced to solve problems with continuous decision variables [10]. The effectiveness of the firefly algorithm has been demonstrated in [11], which stated that it has a better success rate than particle swarm optimization and genetic algorithm. The algorithm was modified to solve discrete problems, including scheduling problems [12]–[14], travelling salesman problems [15], [16], and VRPs [17], [18]. The discrete version of the algorithm was shown to outperform ant colony optimization [12]. The firefly algorithm can effectively search for new solutions because of the automatic subdivision process [19]. During their movements, the fireflies are attracted to other brighter fireflies and form groups of fireflies around local optimal points in the search space. Reactive tabu search (RTS) that allows dynamically changing the tabu list size has also been demonstrated to solve VRP effectively [20]. The tabu list size is updated based on the frequency a solution is found during the search. An adaptive tabu search mechanism that stores good solutions in an elite solution list was used in [21] to solve a capacitated VRP effectively. In our study, we combine both approaches to produce an effective algorithm. Our study introduces some new operators in the proposed firefly and ARTS algorithms.

Our paper is organized as follows: Section II reviews the previous studies. Section III formally defines the problem. Section IV presents an MILP model. Section V shows the developed algorithms. Section VI establishes numerical experiments. Finally, Section VII provides conclusions.

II. LITERATURE REVIEW

In the problem, the drone movements are organized to visit the points and arcs in the network. Some drone routing studies only focus on visiting points. Mufalli *et al.* [22] studied a drone routing problem, in which the type of sensor (e.g., infrared cameras, video recording devices, and radiation detectors) for each drone is selected, and the surveillance benefit for the entire fleet is maximized. Each sensor provides different levels of benefits for the observation but can also result in additional travel time reduction due to its weight. A mathematical programming model and three heuristics were proposed for the problem. Yakıcı [23] addressed a problem of assigning drones to depots and routing them to maximize the total score obtained from the visited points. An integer linear program and an ant colony metaheuristic approach were developed. Coelho *et al.* [24] discussed a drone routing problem to collect and deliver packages while considering seven different objective functions. An MILP model was developed and solved in a metaheuristic framework. Chowdury *et al.* [25] studied a commodity transportation problem in a disaster-affected region, in which optimal distribution center (DC) locations with their service regions and ordering quantities of the DCs were determined while minimizing the total distribution cost. The required holding reorder costs are calculated given the ordering quantities. A probability was given to each demand node to assess the accessibility of each road after the disaster. Murray and Chu [26] discussed a delivery problem, in which trucks and drones collaboratively distribute parcels. MILP formulations and heuristics were developed and tested. Ha *et al.* [27] studied another parcel delivery problem using trucks and drones, in which the total transportation and waiting times of the vehicles were minimized. The problem was formulated mathematically and solved using a local search-based algorithm and a greedy randomized adaptive search procedure (GRASP). Alotaibi *et al.* [28] studied a drone routing problem in a threated area, in which some targets require visits. The objective was to maximize the number of visited points while satisfying the limit of travel time and allowed threat exposure for each drone. Some point candidates were generated to avoid the threats, and a branch-and-cut-and-price algorithm was proposed.

Studies that consider observations or travels through the arcs of a network are classified into arc routing problems. Damodaran *et al.* [29] conducted a study about a hierarchical Chinese postman problem and proposed a method to determine lower bounds in a short time. Prins *et al.* [30] developed a tour splitting algorithm to divide a giant tour of all customers into shorter trips to solve a capacitated arc routing problem. A GRASP and an iterated local search were constructed to solve the problem. Liu *et al.* [31] discussed a capacitated arc routing problem that minimizes the total travel costs. A memetic algorithm with local search was developed for the problem. Chow [32] studied a drone routing problem, which requires multiple visits on arcs. A deterministic arc-inventory routing model was formulated, and an approximate dynamic programming algorithm based on the Monte Carlo simulation method was developed to solve the problem. Our study differs from Chow [32] in that each arc is monitored once to assess its current condition in our problem, whereas each arc was observed continuously during each time interval in Chow [32]. The drone routing problem differs from general VRP in that it must consider battery-restricted travels, payload weight impact on battery consumption [33], and possible travels at more than one altitude. The battery or travel time limit levels require measurements to assure the feasibility of the travels. The factor of battery is also considered in an

electric vehicle transportation system [34]. Different from all studies above, our study considers drone visits to points and arcs simultaneously.

To the best of the authors' knowledge, studies about drone observation at more than one altitude are few. Symington *et al.* [35] conducted an actual experiment using drones to detect a target from various altitudes. They showed that the altitude affects the target presence detection result. Goodrich *et al.* [36] developed a computer vision algorithm to obtain a stabilized image using photos taken from various altitudes by a drone. Waharte and Trigoni [8] suggested the altitude limit at which drones fly. The drones started performing the observation from the highest possible altitude and then reduced the altitudes to refine the target presence estimation. They proposed heuristic algorithms including greedy and Markov-based heuristics to perform search and rescue operations using drones. Zorbas *et al.* [37] studied an optimal drone placement to minimize the cost while assuring the complete surveillance of all targets. An integer linear and mixed integer nonlinear optimization models were formulated, and some centralized and localized heuristics were presented. Zhen *et al.* [9] studied an observation problem using multiple drones at more than one altitude. Zhen *et al.* [9] performed observations for the square area. The proposed method might be unfit when a high-quality observation is required for a long target arc because the arc might be excluded in the defined area. In addition, if the target nodes and arcs are located sparsely, too many areas that can cause extensive computational time must be defined. By contrast, we observe specific target nodes and edges. A target edge can be observed by a drone that travels through an arc located right above the target edge. Our study handles a multiple altitude drone routing problem.

Our study is the first to examine a multiple-drone routing problem at multiple altitudes for node and edge observation while considering the required observation quality and drone battery replacement that minimizes the makespan of drone travel.

III. PROBLEM DEFINITION

During a post-disaster period, supplies must be delivered from distribution centers to residential areas of the refugees [38]–[40]. Accordingly, surveillance is conducted by taking photos using several drones to assess the availability of the supplies at the distribution centers, the magnitude of the disaster at residential areas, the number of people in those areas, and the possibility to use the roads and intersections. A network (S, E) consists of $S = \{1, 2, \ldots, s\}$ and $E = \{1, 2, \ldots, e\}$ that represent sets of target nodes and edges at a disaster area requiring observation. The nodes in S are distribution centers, residential area, and road intersections, whereas the edges in E are roads connecting the nodes. The target nodes and edges must be observed by capturing their photos. Various levels of photo quality are considered in our study. In previous studies, two levels of observation quality [8] and more than two levels [9] were considered. High-quality photos of important target nodes and edges must be captured, such as distribution centers, large residential areas, main roads, and important road intersections. Conversely, less important ones can be sufficiently observed using low-quality photos. During the observation, the required photo quality for each target node and edge must be satisfied.

The drones perform observations while traveling on a network (N, A), where N is the set of the depot and points, and A is the set of arcs. A drone's route consists of one or more subroutes. When two altitudes are considered, each subroute starts and ends at the depot represented with 0 and $2s + 1$, respectively. The sets of drone observation points at the low and high altitudes are $S_{low} = \{1, 2, \ldots, s\}$ and $S_{high} = \{s+1, s+2, \ldots, 2s\}$, respectively. Observation points in S_{low} and S_{high} are located at the same horizontal positions with target nodes on the ground. S_{low} and S_{high} sets form a set of observation points ($S_{observe}$). A set of arcs (A) consists of a set of arcs for observation at low altitude (A_{low}), a set of arcs for observation at high altitude (A_{high}), arcs connecting both depot indices, arcs connecting depot indices with each observation point, and arcs connecting observation points at different altitudes.

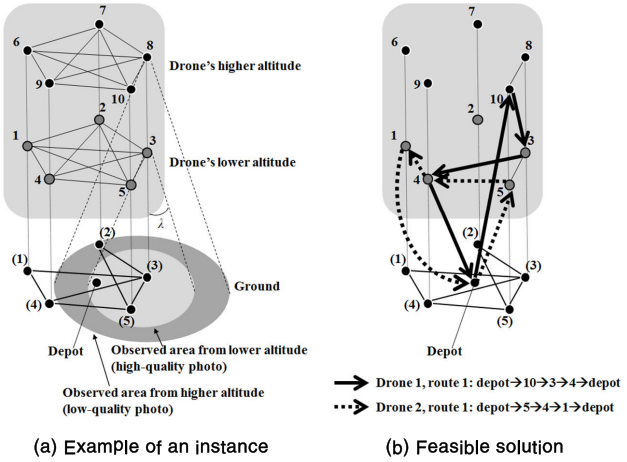

FIGURE 1. Instance and a feasible solution.

Fig. 1 shows an example of an instance and a feasible solution. An observation from a relatively low altitude allows the drones to observe less number of target nodes and edges, whereas drone movements at a relatively high altitude enable the drones to capture photos of more target nodes and edges. For example, Fig. 1(a) shows that Nodes 3 and 5 and Edge (3,5) can be observed from Point 3, whereas a wider area, which contains Nodes 2, 3, and 5 and Edges (2,3), (2,5), and (3,5) can be observed from Point 8, when assuming that the required observation quality of those nodes and edges is satisfied. Some target edges cannot be observed from any point. In that case, drones must fly through arcs above the target edges to perform the observation. Each flight of a drone is limited by its battery level. The drone can continue performing its next flight after returning to the depot to replace its battery. The makespan of the observation is minimized

while considering the battery status of the drones. Fig. 1(b) presents a solution example consisting of two drone routes.

Our study considers the following assumptions:

1) All target nodes and edges require observation at least once with the requested observation quality.
2) The time required to capture a photo is negligible.
3) Travel times between points in the drone network are deterministic (wind effects are not considered).
4) No photo storage limits are set for the drones.

The drone observation problem can be stated as follows:

- Input (given data)
  - (a) Sets of target nodes and edges are provided. The required photo quality for each node or edge is known.
  - (b) A set of drones is provided. Each drone has the same initial full battery level and can have the same number of maximum subroutes.
  - (c) A set of directed arcs that can be travelled by the drones is provided. The required travel time through each arc and the reduced battery level after performing travel through each arc are known.
  - (d) The time required for battery replacement at the depot is provided.
  - (e) Sets of nodes and edges that can be observed at each point are provided.
  - (f) A set of edges that can be observed during drone travel through an arc is provided.
- Output (decision)
  - (a) Route of each drone is determined.
  - (b) Arrival time of each drone in each travel at each depot or point is determined.
  - (c) Visited points or travelled arcs that cover each target node and edge are determined.
  - (d) Battery level of each drone in each travel at each depot or point is determined.
- Goal (objective)

The total observation time (makespan) must be minimized.

IV. MILP MODEL

An MILP formulation is developed for the problem. The parameters are as follows:

c	number of flight altitudes
d	number of available drones
l	maximum possible number of subroutes per individual drone
s	number of target nodes
e	number of target edges
a_{ij}	required time for a drone to travel from stop (depot or point) i to stop (point or depot) j
b	battery replacement time at the depot
$f_{0,k}$	initial battery level of a drone for subroute k at depot 0
g_{ij}	reduced battery level for movement from stop (depot or point) i to stop (point or depot) j

The sets are as follows:

R	set of individual drones, $R = \{1,2,\ldots, d\}$
K	set of all drone subroutes, $K = \{1,2,\ldots, D(= l \times d)\}$
S	set of target nodes, $S = \{1,2,\ldots, s\}$
E	set of target edges, $E = \{1,2,\ldots, e\}$, $(q, y) \in E$
N_o	set of observation points at all altitudes, $N_o = \{1,2,\ldots, B(= c \times s)\}$
N	set of stops (depot and possible observation points), $N = \{0, B+1\}+N_o$
A	set of directed arcs for observation at all altitudes
O_i	set of target nodes covered (with the required quality) when a drone performs the observation at point i, $i \in N_o$, $O_i \subseteq S$
P_i	set of target edges covered (with the required quality) when a drone performs the observation at point i, $i \in N_o$, $P_i \subseteq E$
π_{ij}	set of target edges covered (with the required quality) when a drone performs the observation at arc(i, j), $(i, j) \in A$, $\pi_{ij} \subset E$

The decision variables are as follows:

u_k	1, if subroute k is used for performing observation; 0, otherwise, $k \in K$
x_{ijk}	1, if a drone travels from stop i to stop j in subroute k; 0; otherwise, $i \in N\backslash\{B+1\}$, $j \in N\backslash\{0\}$, $i \neq j$
z	the latest arrival time of drones at depot $B+1$
t_{ik}	the arrival time of a drone in subroute k at the depot or point i, $i \in N$, $k \in K$
f_{ik}	the battery level of a drone in subroute k at the depot or point i, $i \in N$, $k \in K$
α_h	1, if node h is covered by any drone; 0, otherwise, $h \in S$
γ_{ih}	1, if node h is covered by a drone observation at point i; 0, otherwise, $i \in N_o$, $h \in S$
β_{qy}	1, if edge (q, y) is covered by any drone; 0, otherwise, $(q, y) \in E$
w_{qy}	1, if edge (q, y) is covered by a drone visit to any point (with required quality); 0, otherwise, $(q, y) \in E$
ε_{iqy}	1, if edge (q, y) is covered by a drone observation at point i; 0, otherwise, $i \in N_o$, $(q, y) \in E$
m_{qy}	1, if edge (q, y) is covered by a drone observation through any arc (with required quality); 0, otherwise, $(q, y) \in E$

The formulated MILP is as follows:

$$\min z \tag{1}$$

$s.t.$

$$\gamma_{ih} \geq x_{ijk} \quad \forall i \in N_o,\ j \in N\backslash\{0\},\ k \in K,\ h \in O_i \tag{2}$$

$$\gamma_{ih} \geq x_{jik} \quad \forall i \in N_o,\ j \in N\backslash\{cs+1\},\ k \in K,\ h \in O_i \tag{3}$$

$$\sum_k \left(\sum_{j \in N \setminus \{0\}} x_{ijk} + \sum_{j \in N \setminus \{cs+1\}} x_{jik} \right) \geq \gamma_{ih} \quad \forall i \in N_o, \quad h \in O_i \tag{4}$$

$$\gamma_{ih} = 0 \quad \forall i \in N_o, \; h \in S - O_i \tag{5}$$

$$\alpha_h \geq \gamma_{ih} \quad \forall h \in S, \; i \in N_o \tag{6}$$

$$\sum_{i \in N_o} \gamma_{ih} \geq \alpha_h \quad \forall h \in S \tag{7}$$

$$\varepsilon_{iqy} \geq x_{ijk} \quad \forall i \in N_o, \; j \in N \setminus \{0\}, \; k \in K, \; (q, y) \in P_i \tag{8}$$

$$\varepsilon_{iqy} \geq x_{jik} \quad \forall i \in N_o, \; j \in N \setminus \{cs+1\}, \; k \in K, \; (q, y) \in P_i \tag{9}$$

$$\sum_k \left(\sum_{j \in N \setminus \{0\}} x_{ijk} + \sum_{j \in N \setminus \{cs+1\}} x_{jik} \right) \geq \varepsilon_{iqy} \quad \forall i \in N_o, \quad (q, y) \in P_i \tag{10}$$

$$\varepsilon_{iqy} = 0 \quad \forall i \in N_o, \; (q, y) \in E - P_i \tag{11}$$

$$w_{qy} \geq \varepsilon_{iqy} \quad \forall (q, y) \in E, \; i \in N_o \tag{12}$$

$$\sum_{i \in N_o} \varepsilon_{iqy} \geq w_{qy} \quad \forall (q, y) \in E \tag{13}$$

$$m_{qy} \geq x_{ijk} \quad \forall (i, j) \in A, \; (q, y) \in \pi_{ij}, \; k \in K \tag{14}$$

$$\sum_k \sum_{(i,j);(q,y) \in \pi_{ij}} x_{ijk} \geq m_{qy} \quad \forall (q, y) \in E \tag{15}$$

$$\beta_{qy} \geq w_{qy} \quad \forall (q, y) \in E \tag{16}$$

$$\beta_{qy} \geq m_{qy} \quad \forall (q, y) \in E \tag{17}$$

$$w_{qy} + m_{qy} \geq \beta_{qy} \quad \forall (q, y) \in E \tag{18}$$

$$\alpha_h \geq 1 \quad \forall h \in S \tag{19}$$

$$\beta_{qy} \geq 1 \quad \forall (q, y) \in E \tag{20}$$

$$\sum_{j \in N \setminus \{0\}} x_{0,j,k} = 1 \quad \forall k \in K \tag{21}$$

$$\sum_{i \in N \setminus \{cs+1\}} x_{i,cs+1,k} = 1 \quad \forall k \in K \tag{22}$$

$$\sum_{i \in N \setminus \{cs+1\}} x_{imk} - \sum_{j \in N \setminus \{0\}} x_{mjk} = 0 \quad \forall k \in K, \quad m \in N_o \tag{23}$$

$$\sum_{i \in N \setminus \{0, cs+1\}} \sum_{j \in N \setminus \{0\}} x_{i,j,k} \geq u_k \quad \forall k \in K \tag{24}$$

$$x_{i,j,k} \leq u_k \quad \forall i \in N \setminus \{0, cs+1\}, \; j \in N \setminus \{0\}, \; k \in K \tag{25}$$

$$t_{ik} + a_{ij} \leq t_{jk} + M \left(1 - x_{ijk}\right) \quad \forall i \in N \setminus \{cs+1\}, \quad j \in N \setminus \{0\}, \; k \in K \tag{26}$$

$$z \geq t_{cs+1,k} \quad \forall k \in K \tag{27}$$

$$f_{ik} - g_{ij} \geq f_{jk} - M \left(1 - x_{ijk}\right) \quad \forall i \in N \setminus \{cs+1\}, \quad j \in N \setminus \{0\}, \; k \in K \tag{28}$$

$$t_{cs+1,r+wd} + u_{r+(w+1)d} b \leq t_{0,r+(w+1)d} \quad \forall r \in R, \quad w \in \{0, 1, \ldots, l-1\} \tag{29}$$

$$f_{ik} \geq 0 \quad \forall i \in N, \; k \in K \tag{30}$$

$$\alpha_h, u_k, x_{ijk}, m_{qy}, w_{qy}, \beta_{qy} \in \{0, 1\} \quad \forall h \in S, \quad \forall k \in K, \; \forall i, \; j \in N, \; \forall (q, y) \in E \tag{31}$$

$$\gamma_{ih}, \varepsilon_{iqy} \in \{0, 1\} \quad \forall i \in N_o, \; h \in S, \; (q, y) \in E \tag{32}$$

The objective function (1) minimizes the total time required to perform observations of all nodes and edges. Constraints (2)–(7) check whether target nodes can be observed from each observation point at its altitude. Constraints (8)–(18) confirm whether target edges are observed from each observation point or arc at its altitude. Constraints (2)–(7) and (19) ensure that each node is observed by at least one drone. Constraints (8)–(18) and (20) enforce that each edge is observed by at least one drone. Constraints (19) and (20) restrict that each target node and edge must be observed and the photo quality must be satisfied. Constraints (21) and (22) indicate that each drone should start and complete its travel at the depot. Constraints (23) are the flow conservation constraints. Constraints (24)–(25) identify whether or not subroute k is used to perform the observation. Constraints (26) update the arrival time of drone k at the depot or point j if the drone travels from the depot or point i to the depot or point j. Constraints (27) obtain the completion time of all drone operations. Constraints (28) measure the battery level of drone k after its travel from depot or point i to depot or point j. Constraints (29) ensure that the next departure time of drone k is larger than its previous arrival time at the depot and its required battery replacement time. Constraints (30) assure that the drone battery capacity is not violated. Constraints (31)–(32) are binary variable constraints.

The drone surveillance problem is NP-hard. Its NP-hardness can be easily proven via "proof by restriction." Garey and Johnson [41] state that

> Proof by restriction is the simplest, and perhaps most frequently applicable, of our three proof types. An NP-completeness proof by restriction for a given problem L∈NP consists simply of showing that L contains a known NP-complete problem L as a special case.

If we restrict the drone surveillance problem as follows, then the resulting special case is the same as the traveling salesman problem, which is a well-known NP-complete problem.

$E = \varnothing$
$O_i = \{h\}, \forall i \in N_o$, point i is the only corresponding observing point of target node h
$c = 1$
$d = 1$
$l = 1$
$f_{0,k} =$ large number

0	8	2	4	0	5	2	1	0	3	1	4	0	1	5	0

FIGURE 2. Example of a solution.

V. PROPOSED ALGORITHMS

We propose firefly and ARTS algorithms to solve the problem. Both algorithms represent a solution in a sequence of point indices as shown in Fig. 2. In the following examples, two altitudes are considered. Point 0 represents the depot, Points 1–5 are points located at the lower altitude of the drone network, and Points 6–10 are points located at the higher altitude. Two adjacently placed points form an arc; for example, arc (0, 8) is constructed from points 0 to 8 and is traveled by the drone. For each travel order, the subroutes

of the drones are constructed. For example, Fig. 2 shows the first subroute of drone 1 (0, 8, 2, 4, 11), first subroute of drone 2 (0, 5, 2, 1, 11), second subroute of drone 1 (0, 3, 1, 4, 11), and second subroute of drone 2 (0, 1, 5, 11).

An initial population generation procedure (Algorithm 1) is used in both firefly and ARTS algorithms. The definitions of the notations are the same as those in Section IV.

Initially, *pop* solutions are generated. To generate each solution, we assign points and arcs into subroutes (Steps 3–18), which will eventually be assigned to the drones (Step 19). Points and arcs are greedily assigned to subroutes (Steps 4–17). When there are d drones available, d subroutes are generated simultaneously (Step 3) at each iteration. We consider the balance of total travel times of the subroutes in each d number of subroutes to minimize the makespan. A point or arc is selected based on the additional number of observed target nodes and edges and the required additional travel time to insert them into the subroutes (Step 9). A point or arc insertion has a high probability if more new nodes and edges can be observed and the required additional travel time is less. The values are normalized to assure fair considerations of both factors. The selected point or arc is inserted into a subroute that can minimize the completion time measured between subroutes in the considered group (Step 12). When insertion is no longer possible, we continue filling subroutes in the next group (Step 7). At the end of all possible point and arc insertions, if all nodes and edges are observed, then subroutes are assigned to the drones (Step 19). Prior to assigning the subroutes to the drones, we run **remove_unnecessary_observations** procedure to remove unnecessary visits that cause redundant observations of nodes and edges (Step 18). At the end of Algorithm 1, the feasibility of a generated solution is guaranteed.

In the **remove_unnecessary_observations** procedure, we continuously check whether any point or arc must be visited. Every visited point or arc is checked whether it can be removed. If the removal of a visited point or arc does not affect the solution feasibility; that is, all the target nodes and edges remain covered, and the effect of the removal is calculated. Then, the best removal with the largest travel time reduction is selected. The removal process is repeated until a candidate for removal can no longer be found.

Sections V.A and V.B propose the firefly and ARTS algorithms, respectively. In the algorithms, some operators are used to produce new solutions (e.g., higher altitude point insertion, subroute combining, inversion, removal, and arc-swapping operators) during the search for better solutions. New produced solutions are modified further (through a **solution_modification** procedure) as follows: Step 1: Multiple visits of a point within a subroute are removed. In this problem, a point can be visited at most once in each subroute. This characteristic is represented by variables f_{ik} and t_{ik} that are defined in Section IV. Thus, in Step 1, duplicated points in each subroute are checked. If a point is visited more than once within a subroute, then subsequent occurrences of the point are removed.

Algorithm 1 Initial Solution Generating Algorithm

1 : **for** $a = 1$ to *pop* **do**
2 : initialize: *set of observed target nodes* $(S') = \emptyset$ and *set of observed target edges* $(E') = \emptyset$
3 : generate d empty subroutes from depots $\{0, cs+1\}$
4 : **while** *any_target_node_or_edge_is_unobserved* **do**
5 : insert each unvisited point or arc that can be inserted into any subroute and can observe any new target node or edge into *cand_pointarc*
6 : **if** *cand_pointarc* $= \emptyset$ **then**
7 : proceed to Step 4
8 : **else**
9 : select the best subroute for each point or arc j in *cand_pointarc* assuming it is inserted at the end of the subroute, and calculate the score of j as follows: $score_j = \frac{\left(\frac{\text{additional number of observed nodes or edges after adding } j}{\text{total number of nodes and edges}}\right)}{\left(\frac{\text{makespan in group}}{\text{maximum time length of one travel}}\right)}$
10: calculate the selection probability of each j as follows: $prob_j = \frac{score_j}{\sum_{k \epsilon cand\_pointarc} score_k}$
11: m: = randomly select a point or arc from *cand_pointarc* with the probability of Step 10
12: select the subroute that minimizes the longest subroute duration into which m is inserted
13: add m into selected subroute
14: update the *current battery level* and *total travel time* for the selected subroute, S', and E'
15: **if** S': $= S_t$ **and** E': $= E$ **then** proceed to Step 18
16: **end if**
17: **end while**
18: run **remove_unnecessary_observations** procedure
19: assign nonempty subroutes to the drones using the Longest Processing Time rule for $Pm \,||C_{\max}$ problem in [42]
20: **end for**

Step 2: Redundant observations of any target node or arc are removed using **remove_unnecessary_observations** procedure. Step 3: Battery capacity violation in any subroute is resolved by removing the last point (repeatedly if needed) at the end of each subroute. Step 4: If any target node or edge remains unobserved, additional points or arcs are added (Steps 4–17 in Algorithm 1). Step 5: Subroutes are reassigned to the drones using the Longest Processing Time rule for the $Pm \,||C_{\max}$ problem in [42].

At the end of the **solution_modification** procedure, a feasible solution may be produced. If a feasible solution cannot be produced after applying the operators, then the process of generating the new solutions is repeated until a feasible solution is obtained.

A. FIREFLY ALGORITHM

In the algorithm, multiple fireflies simultaneously search within the solution space to effectively determine

Algorithm 2 Higher Altitude Point Insertion Operator

1 : insert each point, which is not at the lowest altitude and appears more times in the reference firefly j than the current firefly, into a set of points ($NLA1$)
2 : **while** $NLA1$ is not empty **do**
3 : **for** each point l in $NLA1$**do**
4 : PO_l: = set of points and arcs in firefly i that become unnecessary to visit if point l is inserted into firefly i
5 : num_l: = number of points in PO_l
6 : $time_l$: = reduced travel time after replacing PO_l with l
7 : **if** num_l = **then** remove l from $NLA1$
8 : **end for**
9 : **if** $NLA1$ is empty **then** break
10: calculate the score of each point l in $NLA1$ as follows:

$$score_l = \frac{\left(\frac{num_l}{\text{total number of points in firefly } i \text{ excluding depots}}\right)}{\left(\frac{time_l}{\text{total travel time of firefly } i}\right)}$$

11: calculate the selection probability of each point l as follows: $prob_l = \frac{score_l}{\sum_{o \epsilon NLA1} score_o}$
12: n: = randomly select a point in $NLA1$ with probability
13: remove n from $NLA1$
14: replace PO_n in firefly i with n
15: **end while**

Algorithm 3 Subroute Combining Operator

1 : given $SUB_{ik} = k^{\text{th}}$ subroute in firefly i, $SUB_{jk} = k^{\text{th}}$ subroute in firefly j, $ALLSUB_i = \bigcup_{k=1}^{D} SUB_{ik}$, $ALLSUB_j = \bigcup_{k=1}^{D} SUB_{jk}$, and $ALLSUB = ALLSUB_i + ALLSUB_j$, calculate the score of each subroute m in $ALLSUB$ as follows:

$$score_m = \frac{\left(\frac{\text{number of observed nodes or edges in subroute } m}{\text{most number of observed nodes and edges in one subroute of fireflies } i \text{and } j}\right)}{\left(\frac{\text{total travel time of subroute } m}{\text{longest total travel time of one subroute in fireflies } i \text{ and } j}\right)}$$

2 : calculate the selection probability of each subroute m as follows: $prob_m = \frac{score_m}{\sum_{o \epsilon ALLSUB} score_o}$
3 : suppose that $NEWSUB_k$ is k^{th} subroute of a new route, $k = 1, \ldots, D$
4 : **while** any $NEWSUB_k$ is empty **do**
5 : n: = randomly select a subroute in $ALLSUB$ with probability
6 : m: = index of subroute n in firefly i or j
7 : **if** selected subroute n belongs to reference firefly j **or** ($NEWSUB_m$ is empty **and** the selected subroute n belongs to reference firefly i) **then**
8 : $NEWSUB_m$: = selected subroute n
9 : **end if**
10: remove the selected subroute n from $ALLSUB$
11: **end while**

good solutions. The three main rules considered in the algorithm [17], [18] are as follows:

- All fireflies are unisex; thus, a firefly can be attracted to another firefly regardless of its sex.
- The attractiveness level of a firefly is determined by its brightness. For any pair of fireflies, the less bright firefly will fly toward the brighter one. The attractiveness level is decreased as the distance between fireflies increases. If no brighter firefly exists, then the considered firefly flies randomly.
- The brightness of a firefly is affected by the landscape of the objective function of the considered problem.

Fireflies move based on some improvements (called movements on most firefly research papers) and mutation operators [15]. Improvement operators are used to generate new better solutions by moving the selected firefly toward a reference firefly. A firefly j is selected as a reference for firefly i if firefly j has a better objective value than firefly i. Mutation operators are used to modify a solution when no reference firefly can be found.

We propose two types of improvement operators (higher altitude point insertion and subroute combining) and two types of mutation operators (inversion and removal). Each operator is thoroughly explained as follows. In the higher altitude point insertion operator (Algorithm 2), points (not at the lowest altitude in reference firefly j) are selected to replace some points in firefly i if the same target nodes and edges can be observed and the subroute travel time is reduced. In the subroute combining operator (Algorithm 3), we copy good subroutes of fireflies i and j to produce a new solution. Good subroutes observe an additional number of target nodes and edges but have shorter total travel times than the other subroutes. In the inversion operator, two random positions in a firefly are selected, and then the points between the selected positions are reversed. In the removal operator, we remove points from a firefly that cause a larger travel time reduction and observe less number of target nodes and edges. The number of points to remove is randomly chosen between [1, length of the firefly]. In each operator, we normalize the travel time or the number of the target node and edge factors when calculating the scores of point, arc, or subroute candidate during the selection. For the removal operator, given j as position index of each point in the firefly (excluding the depots), the score of each point at position j is calculated as $score_j$, shown at the bottom of the previous page, $score_j$ is set to 0 if the denominator equals to

$$score_j = \frac{\left(\frac{\text{reduced travel time if point in position } j \text{ is removed}}{\text{longest total travel time of one subroute in firefly } i}\right)}{\left(\frac{\text{number of observed nodes or edges from point } j, \text{ arc } (j-1, j), \text{ and arc } (j, j+1) \text{ if exist}}{\text{most number of observed nodes and edges in one subroute of firefly } i}\right)}$$

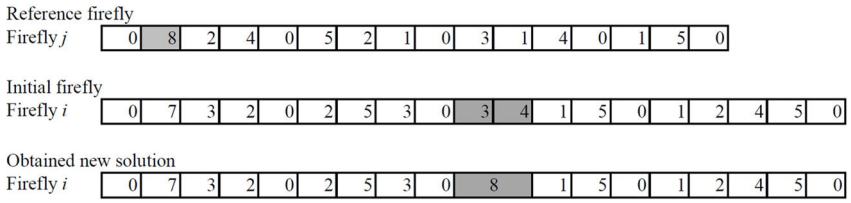

FIGURE 3. Example of higher altitude point insertion operator.

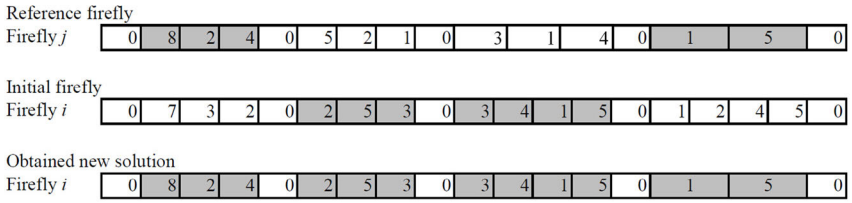

FIGURE 4. Example of subroute combining operator.

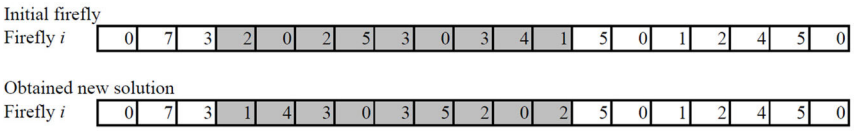

FIGURE 5. Example of inversion operator.

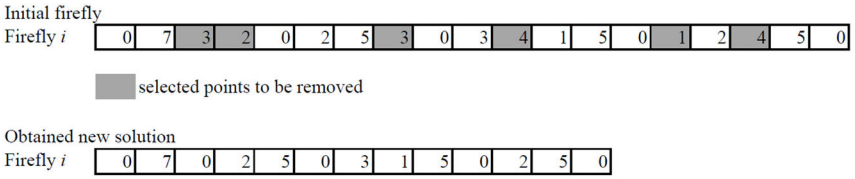

FIGURE 6. Example of removal operator.

0. The probability for removing point j is calculated based on $score_j$. Figs. 3–6 show examples of improvement and mutation operators.

After performing an operator, the **solution_modification** procedure is executed to produce a feasible solution. Algorithm 4 presents the proposed firefly algorithm.

A firefly represents a feasible solution. In the firefly algorithm, many solutions that represent the movements of multiple fireflies are generated. The movements of a number of fireflies (*pop*) are considered simultaneously within an iteration. In the proposed firefly algorithm, the *Tc* number of iterations is considered.

Initially, we generate fireflies for the initial population (Step 1). In each iteration, we consider the movement of each firefly i to find better solutions. During its movement, the firefly searches for a reference firefly (the brightest firefly) with a better objective value of the problem (Step 5) and moves toward it (by applying an improvement operator) (Steps 7–11). If no reference firefly can be found, then the firefly moves randomly through the application of a mutation operator (Steps 13–18).

Prior to searching for a reference firefly j for firefly i, an improvement operator is randomly selected among the higher altitude point insertion and subroute combining operators (Step 4). Among all fireflies with better makespan than firefly i, we select a reference firefly with the highest attractiveness (Step 5). The attractiveness between both fireflies is calculated using Equations (33)–(36).

$$r = \frac{p'}{np'} \times 10 \tag{33}$$

$$\bar{r} = \frac{p + a}{np + na} \times 10 \tag{34}$$

$$\theta = \theta_0 e^{-\delta r^2} \tag{35}$$

$$\bar{\theta} = \theta_0 e^{-\delta \bar{r}^2} \tag{36}$$

Equations (33) and (34) are used to calculate the distances between a reference firefly j and firefly i when higher altitude

Algorithm 4 Firefly Algorithm

pop	size of population
Tc	number of iterations
v	number of moves per firefly
U_i	set of fireflies generated from firefly *i*
f_best	objective value of the best solution
max_same_sol	allowed number of the same consecutive solutions

1 : generate *pop* initial population of fireflies using Algorithm 1 $(i = 1, 2, \ldots, pop)$
2 : **for** $a = 1$ to *Tc* **do**
3 : **for** $i = 1$ to *pop* **do**
4 : insert firefly *i* into U_i *selected firefly*: = *null* $select_{move}$: = a randomly selected improvement operator (with equal selection probability for each operator)
5 : run **select_the_reference_firefly** *j* for firefly *i*
6 : **if** firefly $j \neq null$ **then**
7 : **while** $n(U_i) < v+1$ **do**
8 : *newFirefly*: = a firefly produced by moving firefly *i* to firefly *j* using $select_{move}$
9: apply **solution_modification** procedure to *newFirefly*
10: **if** *newFirefly* is feasible **then** insert the solution into U_i
11: **end while**
12: **else**
13: $select_{mutate}$: = a randomly selected mutation operator
14: **while** $n(U_i) < v+1$ **do**
15: *newFirefly*: = a firefly produced by mutating firefly *i* using $select_{mutate}$
16: apply **solution_modification** procedure to *newFirefly*
17: **if** *newFirefly* is feasible **then** insert the solution into U_i
18: **end while**
19: **end if**
20: select the best firefly in U_i to replace firefly *i*
21: **end for**
22: **if** *f_best* does not change in *max_same_sol* **then** stop
23: **end for**

point insertion and subroute combining operators are applied, respectively, whereas Equations (35) and (36) are used to calculate the attractiveness level between the fireflies based on the distances. Parameter *np*' refers to [the number of points in firefly *j* that are not in the lowest altitude (*NLA2* points)], *p*' to [the number of *NLA2* points that do not exist in firefly *i*], *r* to [the distance between fireflies *i* and *j*], and θ to [the attractiveness level between fireflies *i* and *j* based on the distance *r*]. Parameter *np* refers to [the number of points in firefly *j*], *na* to [the number of arcs in firefly *j*], *p* to [the number of points that exist in firefly *j* but not in firefly *i*], and *a* to [the number of arcs that exist in firefly *j* but not in firefly *i*]. Variable $\bar{r}$ refers to [the distance between both fireflies] and variable $\bar{\theta}$ to [the attractiveness level between fireflies *i* and *j* based on distance $\bar{r}$].

When calculating the distance of the higher altitude point insertion operator (Equation 33), we only assess the number of *NLA2* points because we insert those *NLA2* points when applying the operator. We count the number of points and arcs regardless of their altitudes when calculating the distance of the subroute combining operator (Equation 34) because we deal with all points and arcs when applying the operator.

Parameter θ_0 refers to [the attractiveness level between fireflies *i* and *j* when the distance equals to 0] and δ to [the light absorption coefficient]. The attractiveness is reduced when the distance enlarges. The value of δ affects the attractiveness value of a firefly and the convergence speed of the algorithm. Its value highly depends on the type of problem to be optimized [43]. Multiplication with a constant in Equations (33) and (34) causes the distance values to be within [0,10]. The value of δ is tested between [0.01, 0.15] to set the attractiveness level of a firefly viewed from other fireflies to follow the curve shown in Fig. 7 [15].

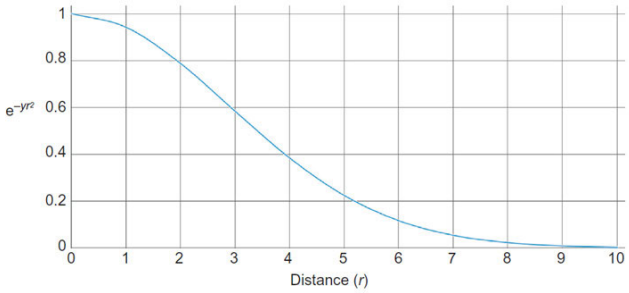

FIGURE 7. Relationships between distances and attractiveness of fireflies when δ is set equal to 0.06.

In Steps 7–11, a fixed number of new fireflies are generated from the movements of a single firefly [15]. In Step 8, we produce new fireflies using the selected improvement operator in Step 4. Given the new firefly, a feasible solution is produced using the **solution_modification** procedure.

If no reference firefly is found in Step 5, then we apply the mutation operator to firefly *i* in Steps 13–18 to produce *v* new fireflies. After producing new fireflies using the improvement or mutation operators, we select the best firefly in U_i to replace firefly *i* (Step 20). We continue the search until the *Tc* number of iterations is reached or the best objective value is not improved after several iterations (Step 22).

B. ARTS ALGORITHM

We propose an arc exchange-based ARTS (Algorithm 5) that dynamically changes the tabu list size (*tls*) using an RTS mechanism [20] and keeps track of the best solutions during the search using an adaptive mechanism [21]. In the RTS [20], *tls* is dynamically changed based on the quality of the obtained solution. *tls* is increased when the same solutions are repeated often and decreased to avoid conducting searches

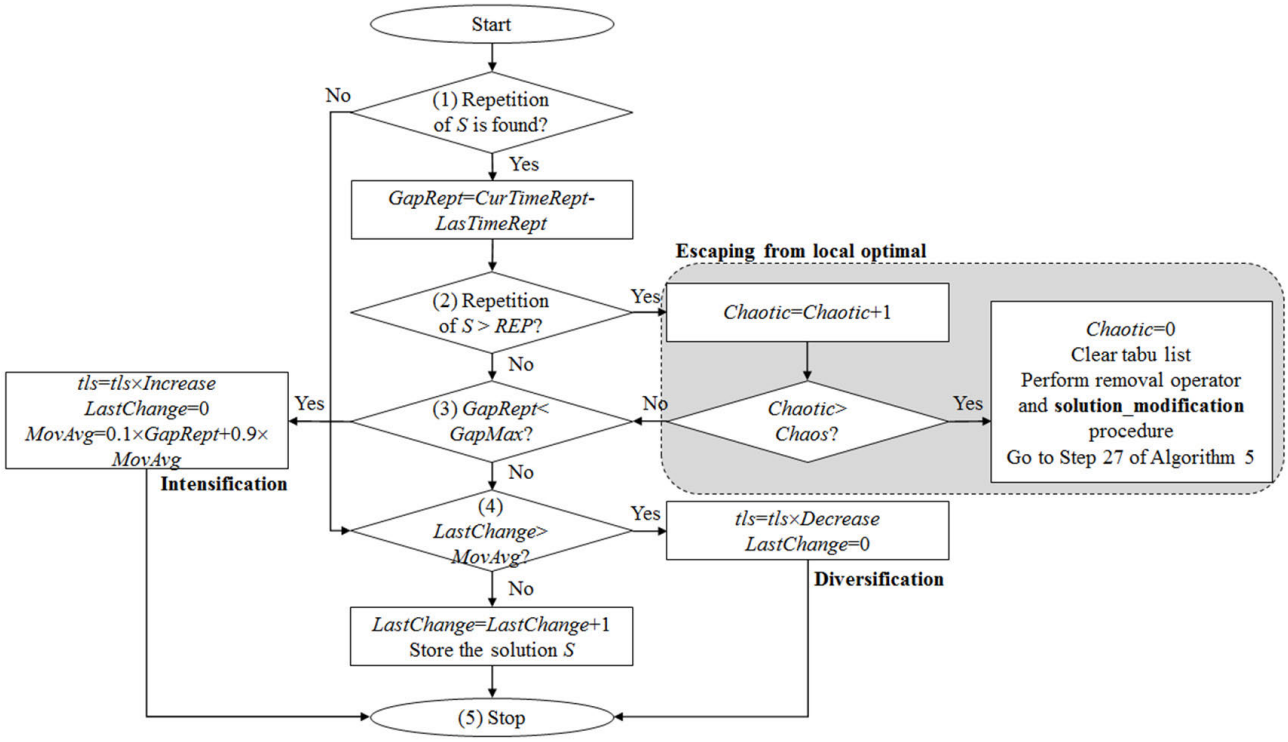

FIGURE 8. RTS mechanism.

only within a certain space. Gounaris *et al.* [21] used the adaptive mechanism to identify a part of a solution that often appears as a good solution. We keep track of good solutions in an *elite_list*. When the best solution is not improved after a certain number of iterations, a new solution is randomly generated by combining two randomly selected solutions from the *elite_list*.

The initial *tls* is calculated as $\alpha \times$ *combination* given that $A =$ number of arcs in the solution and *combination* $= {}_AC_2$. During the search, *tls* is dynamically updated using the RTS mechanism after a search iteration is completed (Step 26) as explained in Fig. 8 [20]. Notations used in Fig. 8 are:

- *Chaotic*: counter for the often-repeated sets of solutions (initialized to 0)
- *MovAvg*: moving average for the detected repetitions (initialized to 0)
- *GapRept*: gap between two consecutive identical solutions (initialized to 0)
- *LastChange*: number of iterations since the last change in *tls* value (initialized to 0)
- *LasTimeRept*: iteration number when last time an identical solution was noticed
- *CurTimeRept*: iteration number of the most recent repetition (initialized to 0)
- *REP*: maximum limit for the often-repeated solutions
- *Chaos*: maximum limit for the sets of often-repeated solutions
- *Increase*: percentage increase for the *tls*
- *Decrease*: percentage decrease for the *tls*
- *GapMax*: constant used to compare with *GapRept* to get the moving average

To reduce the time for checking repetitions of a solution in the RTS mechanism, we store all obtained solutions and objective values in a hash table following the implementation in [44]. If the same solution S is found consecutively within a short number of iterations ($GapRept < GapMax$), then we increase *tls*. Conversely, if *tls* is changed in a sufficiently large number of iterations ($LastChange > MovAvg$), *tls* is decreased. To avoid performing the search on only a certain limited search space, when solution S is found too often (Repetition of $S > REP$ and $Chaotic > Chaos$), we clear the tabu list and produce a new solution to restart the search using the removal mutation. In addition to [20], we add our removal mutation in Fig. 8. The value of *tls* is also changed proportionally to the remaining computational time (Step 8). As the computational time reaches the time limit, *tls* becomes 0. The final iterations of the search focus on intensification rather than exploration by decreasing the value. Every time *tls* is updated in Steps 8 and 26 or a pair is added into the list in Steps 12 and 17, we remove the earliest pairs from the list if the number of pairs in the list is more than *tls*.

An adaptive mechanism is added by maintaining a list of good (elite) solutions. The list is updated using **update_elite_list** procedure (Step 27) [21] as follows: Let R denote a solution set that is a candidate for insertion into the elite solution list (*elite*), R' be any reference solution of

Algorithm 5 ARTS Algorithm

$f(v)$	objective value of solution v
I	set of arc positions in the solution
$n(I)$	size of I
point(i)	arc at position i in the solution
tls	size of tabu list
time_limit	given computational time limit
max_same_sol	allowed number of the same consecutive solutions

1 : generate a set of solutions using Algorithm 1, and select a solution (v) with the smallest $f(v)$
2 : initialize *v_localbest*: $= v$ *v_globalbest*: $= v$ *f_localbest*: $= f(v)$, *f_globalbest*: $= f(v)$
3 : **while** true **do**
4 : initialize *new_localbest*: $= 0$, *v_nontabubest*: $= \emptyset$ *f_nontabubest*: $=$ a large number
5 : **for** $i =$ to $n(I) - 1$ **do**
6 : **for** $j = i + 2$ to $n(I)$ **do**
7 : **if** arc(i) $=$ arc(j) **then** continue
8 : reduce *tls* based on the remaining time
9 : **if** the computational time exceeds *time_limit* **then**
go to Step 22
10: v' $=$ swap(i, j) in v and run **solution_modification** procedure for v'
11: **if** v' is feasible **and** $f(v') \leq f(v)$, **then**
12: add (arc(i), arc(j)) into *tabu list*
13: *v_localbest*: $= v$' and *f_localbest*: $= f(v')$ *new_localbest*: $= 1$
14: **if** $f(v') < f\_globalbest$ **then** set *v_globalbest*: $= v'$, *f_globalbest*: $= f(v')$
15: **end if**
16: **if** (arc(i), arc(j)) is not in the *tabu list* **and** v' is feasible **then**
17: add (arc(i), arc(j)) into *tabu list*
18: **if** $f(v') \leq f\_nontabubest$ **then** *v_nontabubest*: $= v'$, *f_nontabubest*: $= f(v')$
19: **end if**
20: **end for**
21: **end for**
22: **if** *new_localbest* $= 1$ **then** set v: $=$ *v_localbest* $f(v)$: $=$ *f_localbest*
23: **else if** *v_nontabubest* $\neq \emptyset$ set v: $=$ *v_nontabubest* $f(v)$: $=$ *f_nontabubest*
24: **end if**
25: **if** the computational time exceeds *time_limit* **then** stop
26: update *tls* using **RTS mechanism**
27: run **update_elite_list** procedure
28: run **restart_with_new_solution** procedure
29: **if** *f_globalbest* does not change in *max_same_sol* **then** stop
30: **end while**

P, and R^B and R^W be the best and the worst of solutions of P, respectively. Given that $f(R)$ refers to the objective value of solution R, if $f(R) < f(R^B)$, then R replaces R^W in P. Otherwise, if R' satisfying $f(R) < f(R')$ and diss(R, R^B) $>$ diss(R', R^B) exists, then R replaces R' in P. diss(R, R^B) is a dissimilarity distance between solution R and the current best solution R^B that is measured as the total difference of arc occurrences between arcs that exist in R^B but not in R (broken-pair distance). In the **restart_with_new_solution** procedure (Step 28), we calculate the gap of iterations when any *new_localbest* is found. If the value is larger than *max_between_bests*, then we generate a new solution using the subroute combining operator presented in Algorithm 3.

In the ARTS, an acceptance rule in Ropke and Pisinger [45] similar to the simulated annealing is used to accept worse nontabu solutions for escaping from the local optima (Step 16). A new worse sequence v' is accepted with the probability of $e-(f(v')-f(v))/T$ given the current solution v and temperature T. The temperature T initially equals T_{start} and decreases by $T = T \times \xi$ for each iteration, where ξ is the cooling rate. The value of T_{start} is set to allow the acceptance of a ρ% worse solution to be 50%.

TABLE 1. Network-related parameters (node, edge, and altitude) in data sets.

Data Type	Instances	Number of Target Nodes	Number of Target Edges	ALT[a]	RAD[b]
Artificial	A1–A5	4	3–4	40,80	12.8, 25.6
	A6–A10	5	6–9		
	A11–A15	6	10–13		
	A15–A20	7	12–17		
Actual	R21	22	24	120, 400	38.4, 128.0
	R22	19	23		
	R23	19	18		
	R24	22	21		

[a]represents the considered altitudes (m)
[b]represents the observation radius from the altitudes (m)

VI. NUMERICAL EXPERIMENTS

Artificial and actual data sets with different characteristics are considered (Tables 1 and 2). In the artificial data set, the locations of target nodes and edges, required travel times, and observation qualities are generated randomly. However, for the actual data set, we set those values based on the actual instances. The artificial data set is generated to show the quality of the proposed methods compared with the optimal solutions of the MILP, and the actual data set is utilized to test the effectiveness of the proposed algorithms in solving real-sized problems. In both data sets, λ is set to 60^o as shown in Fig. 1(a) [37].

The required photo qualities for target nodes and edges are generated randomly for the artificial and actual data that are 0 (low-quality) and 1 (high-quality). Target nodes and edges that can be observed from the points are listed through prepro-

TABLE 2. Drone-related parameters in data sets.

Parameter	Artificial Data	Actual Data
Given travel time between target nodes (a_{ij}) (s)	[2,8]	[2.8,57.04][a]
Horizontal speed (m/s)	10[b]	
Upward speed (m/s)	4[b]	
Downward speed (m/s)	8[c]	
(Number of drones, number of subroutes per drone)	(2,2)	(2,3)
Battery replacement time	1	90[d]
Battery limit per subroute	34,38,36,40	-
Battery level reduction per travel time	1 (equivalent with a_{ij})	-
Travel time limit per subroute (s)	-	320,320, 240,270[e]

[a]latitude and longitude of nodes are obtained from Google Map, the horizontal speed is used to calculate the travel times
[b]refer to [46]
[c]assumed on the basis of [9]
[d]refer to [47]
[e]refer to [48]

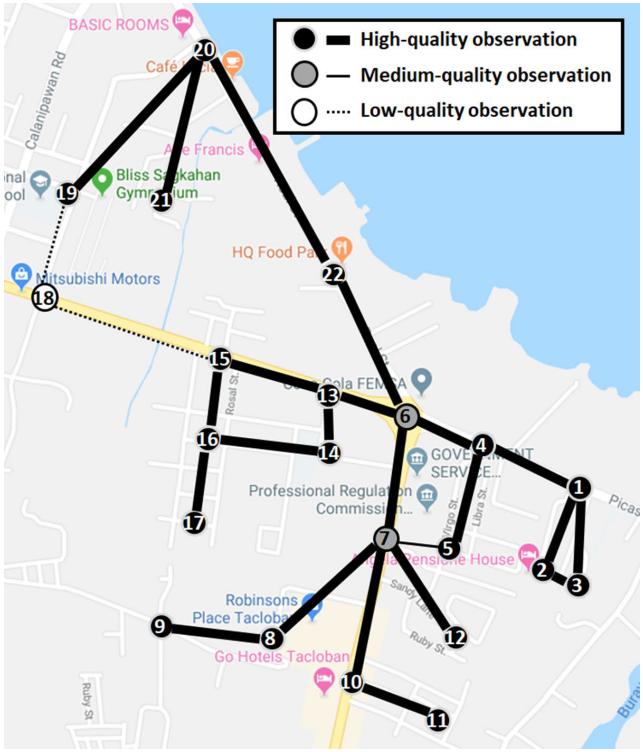

FIGURE 9. Target nodes and edges in the western part of the Philippines.

cessing. A drone that travels through an arc can observe the target edge located below the travelled arc. When generating the drone networks, arcs that cannot be feasibly travelled due to the travel time limit (from and to the depots) are removed.

Actual areas affected by natural disasters, such as the Philippines [49], Mexico [50], and the Republic of Korea [51], are considered the study cases in the actual data set. The Philippines is divided into two. Fig. 9 shows the map for the western part of the Philippines.

The algorithm parameters for the firefly and ARTS are tuned via preliminary experiments on all data sets.

TABLE 3. Tuned parameters for firefly and ARTS algorithms.

Algorithms	Parameter	Artificial Data	Actual Data
Firefly	δ	0.07	0.04
	θ_0	2	2
	v	15	10
	pop	60	20
	Tc	100,000	100,000
	max_same_sol	30	20
ARTS	α	0.7	0.25
	ρ	20	65
	ξ	0.8	0.95
	$size(elite)$	20	20
	$max\_between\_bests$	10	400
	REP	15	15
	$Chaos$	3	6
	$GapMax$	6	15
	$Increase$	1.05	1.25
	$Decrease$	0.5	0.95
	max_same_sol	30	5

Table 3 presents the obtained parameters in the firefly and ARTS algorithms.

The MILP solutions are obtained using CPLEX 12.9.0. All experiments were performed on an Intel®Core™ i5-6400 CPU at 2.70 GHz with 8 GB RAM. Computational time is limited to 1,800 s (for MILP) and 60 s (for each algorithm). The computational time must be as short as possible considering that when disaster occurs, rapid observation allows us to supply disaster relief aids (e.g., through land and air) within a short period of time. Table 4 presents the experiment results. The firefly algorithm and ARTS use the same initial solution set. The best solution in the set is used as the initial solution for MILP1 also. For each instance, each algorithm is executed five times, and the average makespan is obtained. Objective value gaps between the firefly algorithm or ARTS with MILP are defined as $(z_{algorithm} - z_{MILP})/z_{MILP} \times 100$. MILP1 can only obtain the optimal solution in instance A1 and cannot find any feasible solution in most artificial and actual instances.

For the artificial data sets, the proposed algorithms produce better solutions than the MIP1 does, and the average gaps between the firefly algorithm and ARTS with the MILP1 are -3.6% and -0.5%, respectively. The proposed algorithms produce acceptable solutions within a short computation time (less than 60 s). The firefly algorithm and ARTS obtain solutions with an average gap of -22.8% and -29.2%, respectively than those of MILP1, for the actual data instances.

Although ARTS generates slightly worse solutions than the firefly algorithm (35.9 vs. 35.1 in average z value) for the artificial data sets, the computation time of ARTS (1.5 s) is much shorter than that of the firefly algorithm (31.7 s). For the actual data instances, ARTS outperforms the firefly in solution quality and computation time. The average makespan of the solutions generated by ARTS is 442, whereas the one by the firefly is 463. The average computation times of ARTS and the firefly algorithm are 9.0 and 43.1 s, respectively. In summary, ARTS outperforms the firefly algorithm.

TABLE 4. Comparison of MILP (without and with initial solutions), firefly algorithm, and arts algorithms.

Instance no.	MILP0 (MILP without Initial Solution)		MILP1 (MILP with Initial Solution)		Firefly Algorithm					ARTS					Gap (MILP1 and Firefly) (%)	Gap (MILP1 and ARTS) (%)	Gap (Firefly and ARTS)			
	z	Comp time (s)	*z*	Comp time (s)	*z*	*#alt2 (points, arcs)* (%)	*#sub (dr1, dr2)*	*travD* (avg, stdev)	Comp time (s)	*z*	*#alt2 (points, arcs)* (%)	*#sub (dr1, dr2)*	*travD* (avg, stdev)	Comp time (s)			*z* (%)	*#alt2 (points, arcs)*	*#sub (dr1, dr2)*	*travD (avg, stdev)*
A1	21* (7)	1,800	18	138	18	(0.0, 0.0)	(1.0, 1.0)	(16.5, 1.5)	5	18	(0.0, 0.0)	(1.0, 1.0)	(16.5, 1.5)	1	0.0	0.0	0.0	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)
A2	24* (18)	1,800	24* (19)	1,800	24	(0.0, 0.0)	(1.0, 1.0)	(22.5, 1.5)	6	25	(0.0, 0.0)	(1.0, 1.0)	(23.4, 1.8)	1	0.0	5.0	5.0	(0.0, 0.0)	(0.0, 0.0)	(0.9, 0.3)
A3	21* (10)	1,800	21* (11)	1,800	21	(0.0, 0.0)	(1.0, 1.0)	(17.8, 3.2)	5	21	(0.0, 0.0)	(1.0, 1.0)	(17.5, 3.5)	1	0.0	0.0	0.0	(0.0, 0.0)	(0.0, 0.0)	(–0.3, 0.3)
A4	14* (8)	1,800	14* (9)	1,800	14	(0.0, 0.0)	(1.0, 1.0)	(14.0, 0.0)	8	17	(0.0, 0.0)	(1.0, 1.0)	(15.3, 1.3)	1	0.0	18.6	18.6	(0.0, 0.0)	(0.0, 0.0)	(1.3, 1.3)
A5	16* (5)	1,800	16* (5)	1,800	16	(0.0, 0.0)	(1.0, 1.0)	(15.2, 0.8)	8	17	(0.0, 0.0)	(1.0, 1.0)	(15.6, 1.6)	1	0.0	7.5	7.5	(0.0, 0.0)	(0.0, 0.0)	(0.4, 0.8)
A6	-**	1,800	33* (0)	1,800	33	(0.0, 0.0)	(1.0, 1.2)	(31.6, 1.4)	26	34	(0.0, 0.0)	(1.0, 1.0)	(32.8, 0.8)	1	0.0	1.8	1.8	(0.0, 0.0)	(0.0, –0.2)	(1.2, –0.6)
A7	-**	1,800	40* (0)	1,800	35	(0.0, 0.0)	(1.0, 2.0)	(35.1, 0.3)	26	36	(0.0, 0.0)	(1.2, 2.0)	(35.5, 0.7)	1	–11.5	–9.5	2.3	(0.0, 0.0)	(0.2, 0.0)	(0.4, 0.4)
A8	-**	1,800	42* (0)	1,800	41	(0.0, 0.0)	(1.2, 2.0)	(38.1, 3.3)	34	42	(0.0, 0.0)	(1.4, 2.0)	(39.0, 2.8)	1	–1.4	–0.5	1.0	(0.0, 0.0)	(0.2, 0.0)	(0.9, –0.5)
A9	36* (0)	1,800	29* (0)	1,800	29	(0.0, 0.0)	(1.0, 1.4)	(26.8, 2.2)	11	29	(0.0, 0.0)	(1.0, 1.0)	(26.1, 2.9)	1	0.0	0.0	0.0	(0.0, 0.0)	(0.0, –0.4)	(–0.7, 0.7)
A10	-**	1,800	44* (0)	1,800	41	(0.0, 0.0)	(1.0, 1.8)	(40.2, 0.8)	26	42	(0.0, 0.0)	(1.6, 2.0)	(41.2, 0.8)	1	–6.8	–4.5	2.4	(0.0, 0.0)	(0.6, 0.2)	(1.0, 0.0)
A11	-**	1,800	50* (0)	1,800	48	(0.0, 0.0)	(1.0, 2.0)	(43.6, 5.0)	47	49	(0.0, 0.0)	(1.0, 2.0)	(43.5, 4.9)	1	–2.8	–2.8	0.0	(0.0, 0.0)	(0.0, 0.0)	(–0.1, 0.1)
A12	-**	1,800	43* (0)	1,800	42	(0.0, 0.0)	(2.0, 2.0)	(42.0, 0.0)	60	42	(0.0, 0.0)	(2.0, 2.0)	(42.3, 0.1)	1	–2.3	–1.4	1.0	(0.0, 0.0)	(0.0, 0.0)	(0.3, 0.1)
A13	-**	1,800	39* (0)	1,800	35	(0.0, 0.0)	(1.0, 2.0)	(33.4, 2.0)	36	37	(0.0, 0.0)	(1.0, 2.0)	(33.9, 2.7)	1	–9.2	–6.2	3.4	(0.0, 0.0)	(0.0, 0.0)	(0.5, 0.7)
A14	-**	1,800	48* (0)	1,800	47	(0.0, 0.0)	(2.0, 2.0)	(46.6, 0.4)	60	47	(0.0, 0.0)	(2.0, 2.0)	(46.6, 0.4)	2	–2.1	–2.1	0.0	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)
A15	-**	1,800	38* (0)	1,800	33	(0.0, 0.0)	(1.0, 2.0)	(33.0, 0.0)	31	38	(0.0, 0.0)	(2.0, 2.0)	(38.4, 0.0)	1	–13.2	1.1	16.4	(0.0, 0.0)	(1.0, 0.0)	(5.4, 0.0)
A16	-**	1,800	46* (0)	1,800	41	(0.0, 0.0)	(1.4, 2.0)	(40.6, 1.2)	59	43	(0.0, 0.0)	(1.6, 2.0)	(42.2, 0.6)	2	–9.1	–7.0	2.4	(0.0, 0.0)	(0.2, 0.0)	(1.6, –0.6)
A17	-**	1,800	43* (0)	1,800	39	(7.3, 0.0)	(1.0, 2.0)	(39.0, 0.6)	45	40	(7.6, 0.0)	(1.0, 1.8)	(39.1, 0.9)	1	–7.9	–7.0	1.0	(0.3, 0.0)	(0.0, –0.2)	(0.1, 0.3)
A18	-**	1,800	60* (0)	1,800	58	(4.3, 0.0)	(2.0, 2.0)	(57.9, 0.1)	60	58	(7.9, 3.1)	(1.6, 2.0)	(56.9, 0.9)	9	–3.3	–3.7	–0.3	(3.6, 3.1)	(–0.4, 0.0)	(–1.0, 0.8)
A19	-**	1,800	47* (0)	1,800	46	(9.5, 6.0)	(1.0, 2.0)	(43.4, 2.8)	44	45	(7.0, 0.0)	(1.0, 1.8)	(43.9, 1.5)	2	–1.7	–3.4	–1.7	(–2.5, –6.0)	(0.0, –0.2)	(0.5, –1.3)
A20	-**	1,800	37* (0)	1,800	36	(24.4, 15.7)	(1.0, 1.0)	(36.0, 0.6)	37	39	(15.5, 12.5)	(1.2, 1.4)	(37.5, 1.1)	1	–1.1	4.3	5.5	(–8.9, –3.2)	(0.2, 0.4)	(1.5, 0.5)
R1	-**	1,800	831* (0)	1,800	552	(0.0, 0.0)	(2.0, 2.0)	(548, 4.0)	52	566	(0.0, 0.0)	(1.8, 2.0)	(561, 4.7)	15	–33.6	–31.9	2.5	(0.0, 0.0)	(–0.2, 0.0)	(13.4, 0.7)
R2	-**	1,800	738* (0)	1,800	485	(14.9, 13.6)	(2.0, 2.0)	(466, 18.5)	24	488	(7.7, 2.2)	(1.6, 2.0)	(483, 4.8)	8	–34.3	–33.9	0.7	(–7.2, –11.4)	(–0.4, 0.0)	(17.0, –13.8)
R3	-**	1,800	298* (0)	1,800	301	(15.5, 8.1)	(1.2, 1.6)	(285, 16.1)	36	234	(16.3, 5.9)	(1.0, 1.6)	(229, 5.2)	3	1.1	–21.5	–22.4	(0.9, –2.2)	(–0.2, 0.0)	(–56.6 –10.9)
R4	-**	1,800	679* (0)	1,800	513	(0.0, 0.0)	(2.0, 2.0)	(502, 10.5)	60	479	(4.9, 2.2)	(2.0, 2.0)	(476, 3.1)	10	–24.4	–29.4	–6.6	(4.9, 2.2)	(0.0, 0.0)	(–26.5 –7.5)
Average (A1–A20)		1,800	-	1,712	35.1	(2.3, 1.1)	(1.2, 1.6)	(33.7, 1.4)	31.7	35.9	(1.9, 0.8)	(1.3, 1.6)	(34.4, 1.5)	1.5	–3.6	–0.5	3.3	(–0.4, –0.3)	(0.10, –0.02)	(0.70, 0.15)
Average (R1–R4)		1,800	-	1,800	463	(7.6, 5.4)	(1.8, 1.9)	(450, 12.3)	43.1	442	(7.2, 2.6)	(1.6, 1.9)	(437, 4.4)	9.0	–22.8	–29.2	–6.4	(–0.4, –2.9)	(–0.2, 0.0)	(–13.2 –7.9)

*The best feasible solution is obtained within a limited time. (value) shows the best lower bound.
**No feasible solution exists.

Table 4 also provides some Key Performance Indicators (KPIs) for comparing both algorithms as follows: (1) percentage of visited points and arcs at the higher altitude (*#alt2*), (2) number of used drone subroutes (*#sub*) for each drone (*dr1*, *dr2*), and (3) the average (*avg*) and standard deviation values (*stdev*) of the completion time by the

drones (*travD*) that consider the battery replacement times between subroutes. For the first term, we consider that *#alt1* and *#alt2* refer to the number of points or arcs at the lower altitude and the higher altitude, in percentage (*#alt2 points* = 100 × number of points at the higher altitude / number of points in the solution). We list only the *#alt2* values. The last term is used to assess the workload balance of the drones [52] that is related to the makespan minimization. Gaps of the z values are defined as $(z_{ARTS} - z_{Firefly})/z_{Firefly} \times 100$, whereas the gaps of other KPIs are defined as $(z_{ARTS} - z_{Firefly})$. The ARTS obtains solutions with an average makespan gap of 3.3% and −6.4% for artificial and actual data sets, respectively, compared with those of the firefly algorithm. Drones in solutions of both algorithms visit a similar number of points and arcs at higher altitudes as shown by the *#alt2* gap. Firefly has lower average makespan in the artificial data and ARTS in the actual data when each algorithm produces less number of drone subroutes (*#sub*), less average drone completion time (*travD*, *avg*), and less deviation between drone completion times (*travD*, *stdev*).

VII. CONCLUSION

A multiple drone routing problem at multiple altitudes is discussed. The drones perform observations of target nodes and edges that require different photo qualities. In addition, the drones have a limited battery level and must return to depot for a battery replacement process. The objective is to minimize the total observation time. An MILP model, a firefly algorithm, and an adaptive–reactive tabu search (ARTS) algorithms are developed to solve the problem. Experiment results show that the latter outperforms the former algorithms when solving actual data instances. ARTS obtained better solutions within less than 1 min than the MILP that uses the same initial solutions with an average gap of 0.5% and 29.2% for artificial and actual data instances, respectively.

For further studies related to drone routing, a drone network design problem should be investigated. In the drone network design, various decision variables, such as observation altitudes and horizontal positions of the observation points at each altitude, can be considered. Identifying the best observation altitude is also necessary to determine the best observation range and the appropriate altitudes that allow the drones to visit those points while performing the observations in a short time. Possible collisions between the drones must be avoided, especially if the visited points are located close to one another.

REFERENCES

[1] P. B. Sujit and D. Ghose, "Search using multiple UAVs with flight time constraints," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 40, no. 2, pp. 491–509, Apr. 2004.

[2] A. E. Gil, K. M. Passino, and J. B. Cruz, "Stable cooperative surveillance with information flow constraints," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 5, pp. 856–868, Sep. 2008.

[3] M. Simic, C. Bil, and V. Vojisavljevic, "Investigation in wireless power transmission for UAV charging," *Procedia Comput. Sci.*, vol. 60, pp. 1846–1855, Jan. 2015.

[4] K. R. B. S. and A. Poondla, "Performance analysis of solar powered unmanned aerial vehicle," *Renew. Energy*, vol. 104, pp. 20–29, Apr. 2017.

[5] A. Al-Kaff, D. Martín, F. García, A. D. L. Escalera, and J. María Armingol, "Survey of computer vision algorithms and applications for unmanned aerial vehicles," *Expert Syst. Appl.*, vol. 92, pp. 447–463, Feb. 2018.

[6] Y. Liu, Z. Luo, Z. Liu, J. Shi, and G. Cheng, "Cooperative routing problem for ground vehicle and unmanned aerial vehicle: The application on intelligence, surveillance, and reconnaissance missions," *IEEE Access*, vol. 7, pp. 63504–63518, 2019.

[7] V. Rodríguez-Fernández, H. D. Menéndez, and D. Camacho, "Analysing temporal performance profiles of UAV operators using time series clustering," *Expert Syst. Appl.*, vol. 70, pp. 103–118, Mar. 2017.

[8] S. Waharte and N. Trigoni, "Supporting search and rescue operations with UAVs," in *Proc. Int. Conf. Emerg. Secur. Technol.*, Washington, DC, USA, Sep. 2010, pp. 142–147.

[9] L. Zhen, M. Li, G. Laporte, and W. Wang, "A vehicle routing problem arising in unmanned aerial monitoring," *Comput. Oper. Res.*, vol. 105, pp. 1–11, May 2019.

[10] X.-S. Yang, "Efficiency analysis of swarm intelligence and randomization techniques," *J. Comput. Theory Nanosci.*, vol. 9, no. 2, pp. 189–198, Feb. 2012.

[11] X. S. Yang, "Firefly algorithms for multimodal optimization," in *Proc. 5th Int. Symp. Stochastic Algorithms, Found. Appl. (SAGA)*, Sapporo, Japan, 2009, pp. 169–178.

[12] M. K. Sayadi, R. Ramezanian, and N. Ghaffari-Nasab, "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems," *Int. J. Ind. Eng. Computations*, vol. 1, no. 1, pp. 1–10, Jul. 2010.

[13] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 301–305, Apr. 2014.

[14] A. E. Ezugwu and F. Akutsah, "An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times," *IEEE Access*, vol. 6, pp. 54459–54478, Oct. 2018.

[15] G. K. Jati, R. Manurung, and Suyanto, "Discrete firefly algorithm for traveling salesman problem: A new movement scheme," in *Swarm Intelligence and Bio-Inspired Computation*, X. S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, Eds. London, U.K.: Elsevier, 2013, pp. 295–312.

[16] L. Zhou, L. Ding, and X. Qiang, "A multi-population discrete firefly algorithm to solve TSP," in *Bio-Inspired Computing—Theories and Applications*, L. Pan, G. Păun, M. J. Pérez-Jiménez, and T. Song, Eds. Wuhan, China: Springer, 2014, pp. 648–653.

[17] E. Osaba, X.-S. Yang, F. Diaz, E. Onieva, A. D. Masegosa, and A. Perallos, "A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy," *Soft Comput.*, vol. 21, no. 18, pp. 5295–5308, 2017.

[18] C. Wang and X. Chu, "An improved firefly algorithm with specific probability and its engineering application," *IEEE Access*, vol. 7, pp. 57424–57439, 2019.

[19] X. S. Yang and X. He, "Firefly algorithm: Recent advances and applications," *Int. J. Swarm Intell.*, vol. 1, no. 1, pp. 36–50, Aug. 2013.

[20] N. A. Wassan, A. H. Wassan, and G. Nagy, "A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries," *J. Combinat. Optim.*, vol. 15, no. 4, pp. 368–386, 2008.

[21] C. E. Gounaris, P. P. Repoussis, C. D. Tarantilis, W. Wiesemann, and C. A. Floudas, "An adaptive memory programming framework for the robust capacitated vehicle routing problem," *Transp. Sci.*, vol. 50, no. 4, pp. 1239–1260, Jun. 2016.

[22] F. Mufalli, R. Batta, and R. Nagi, "Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans," *Comput. Oper. Res.*, vol. 39, no. 11, pp. 2787–2799, Nov. 2012.

[23] E. Yakıcı, "Solving location and routing problem for UAVs," *Comput. Ind. Eng.*, vol. 102, pp. 294–301, Dec. 2016.

[24] B. N. Coelho, V. N. Coelho, I. M. Coelho, L. S. Ochi, R. Haghnazar K., D. Zuidema, M. S. F. Lima, and A. R. da Costa, "A multi-objective green UAV routing problem," *Comput. Oper. Res.*, vol. 88, pp. 306–315, Dec. 2017.

[25] S. Chowdhury, A. Emelogu, M. Marufuzzaman, S. G. Nurre, and L. Bian, "Drones for disaster response and relief operations: A continuous approximation model," *Int. J. Prod. Econ.*, vol. 188, pp. 167–184, Jun. 2017.

[26] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 86–109, May 2015.
[27] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "On the min-cost traveling salesman problem with drone," *Transp. Res. C, Emerg. Technol.*, vol. 86, pp. 597–621, Jan. 2018.
[28] K. A. Alotaibi, J. M. Rosenberger, S. P. Mattingly, R. K. Punugu, and S. Visoldilokpun, "Unmanned aerial vehicle routing in the presence of threats," *Comput. Ind. Eng.*, vol. 115, pp. 190–205, Jan. 2018.
[29] P. Damodaran, M. Krishnamurthi, and K. Srihari, "Lower bounds for hierarchical Chinese postman problem," *Int. J. Ind. Eng., Theory*, vol. 15, no. 1, pp. 36–44, 2008.
[30] C. Prins, N. Labadi, and M. Reghioui, "Tour splitting algorithms for vehicle routing problems," *Int. J. Prod. Res.*, vol. 47, no. 2, pp. 507–535, Jan. 2009.
[31] T. Liu, Z. Jiang, and N. Geng, "A memetic algorithm with iterated local search for the capacitated arc routing problem," *Int. J. Prod. Res.*, vol. 51, no. 10, pp. 3075–3084, May 2013.
[32] J. Y. J. Chow, "Dynamic UAV-based traffic monitoring under uncertainty as a stochastic arc-inventory routing policy," *Int. J. Transp. Sci. Technol.*, vol. 5, no. 3, pp. 167–185, Oct. 2016.
[33] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 1, pp. 70–85, Jan. 2017.
[34] C. Lee and J. Han, "Benders-and-Price approach for electric vehicle charging station location problem under probabilistic travel range," *Transp. Res. B, Methodol.*, vol. 106, pp. 130–152, Dec. 2017.
[35] A. Symington, S. Waharte, S. Julier, and N. Trigoni, "Probabilistic target detection by camera-equipped UAVs," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, May 2010, pp. 4076–4081.
[36] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini UAV," *J. Field Robot.*, vol. 25, nos. 1–2, pp. 89–110, Jan./Feb. 2008.
[37] D. Zorbas, L. Di P. Pugliese, T. Razafindralambo, and F. Guerriero, "Optimal drone placement and cost-efficient target coverage," *J. Netw. Comput. Appl.*, vol. 75, pp. 16–31, Nov. 2016.
[38] W.-S. Lee, B. S. Kim, and P. F. Opit, "A stock pre-positioning model to maximize the total expected relief demand of disaster areas," *Ind. Eng. Manage. Syst.*, vol. 13, no. 3, pp. 297–303, Sep. 2014.
[39] S. Han, J. Park, and H. Jeong, "A solution procedure for emergency logistics problem in disaster scene," *Ind. Eng. Manage. Syst.*, vol. 17, no. 2, pp. 166–176, Jun. 2018.
[40] R. Patrisina, N. Sirivongpaisal, and S. Suthummanon, "A logistical relief distribution preparedness model: Responses to a probable tsunami case study in west sumatra, indonesia," *Ind. Eng. Manage. Syst.*, vol. 17, no. 4, pp. 850–863, Dec. 2018.
[41] M. R. Garey, and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1979.
[42] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. New York, NY, USA: Springer, 2008, pp. 112–115.
[43] R. Goel and R. Maini, "A hybrid of ant colony and firefly algorithms (HAFA) for solving vehicle routing problems," *J. Comput. Sci.*, vol. 25, pp. 28–37, Mar. 2018.
[44] N. Wassan, "Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls" *J. Oper. Res. Soc.*, vol. 58, no. 12, pp. 1630–1641, 2007.
[45] S. Ropke and D. Pisinger, "An adaptive large neighbourhood search heuristic for the pickup and delivery problem with time windows," *Transport. Sci.*, vol. 40, pp. 455–472, Nov. 2006.
[46] MicroMultiCopter Aero Technology. *Skylle 1550*. Accessed: Jul. 27, 2019. [Online]. Available: http://www.mmcuav.com/portfolio-items/skylle-1550/
[47] V. Golfier, "How to incorporate unmanned aerial vehicles to improve delivery of door-to-door air cargo?" M.S. thesis, Dept. Civil. Eng., ETH Zürich, Zürich, Switzerland, 2018.
[48] Digital World. *Maxfly Camera Drone WiFi XX5W, FPV*. Accessed: Jan. 3, 2020. [Online]. Available: https://topdigital.bg/en/catalogue/drones/maxfly-camera-drone-wifi-xx5w-fpv/
[49] Humanitarian OpenStreetMap Team. (Nov. 10, 2013). *Disaster Activation: Typhoon Haiyan 2013, HOT*. Accessed: Jul. 17, 2019. [Online]. Available: https://www.hotosm.org/projects/typhoon_haiyan
[50] N. Chavez. (Sep. 20, 2017). *Central Mexico Earthquake Kills More Than 200, Topples Buildings, CNN*. Accessed: Jul. 17, 2019. [Online]. Available: https://edition.cnn.com/2017/09/19/americas/mexico-earthquake/index.html
[51] H.-S. Kim, C.-G. Sun, and H.-I. Cho, "Geospatial assessment of the post-earthquake hazard of the 2017 pohang earthquake considering seismic site effects," *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 9, p. 375, Sep. 2018.
[52] X. Yan, B. Xiao, Y. Xiao, Z. Zhao, L. Ma, and N. Wang, "Skill vehicle routing problem with time windows considering dynamic service times and Time-Skill-Dependent costs," *IEEE Access*, vol. 7, pp. 77208–77221, 2019.

IVAN KRISTIANTO SINGGIH (Member, IEEE) received the B.S. and M.S. degrees in industrial engineering from the Bandung Institute of Technology, Indonesia, in 2009 and 2010, respectively, and the Ph.D. degree in industrial engineering from Pusan National University, Busan, South Korea, in 2017. He was a Postdoctoral Researcher with the Department of Industrial and Management Engineering, POSTECH. He is a Research Assistant Professor with the Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), South Korea. His research interests include container terminal logistics, vehicle routing problems, scheduling, and serious game development. He is a member of KIIE and T-LOG. He received the Best Student Paper Award at the IJIE 2013 Conference and the Korean Government Scholarship Excellent Academic Achievement Award in 2014.

JONGHWA LEE received the B.S. degree in industrial and management engineering from the Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 2017, where he is currently pursuing the Ph.D. degree in industrial and management engineering. His research interests include combinatorial optimization and logistics transportation.

BYUNG-IN KIM received the B.S. and M.S. degrees in industrial engineering from the Pohang University of Science and Technology (POSTECH), South Korea, in 1991 and 1994, respectively, and the Ph.D. degree in decision sciences and engineering systems from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2002. He was an Assistant Professor of industrial and systems engineering with the University of Memphis and the Director of Research and Development with the Institute of Information Technology, Inc., The Woodlands, TX, USA. He is currently a Professor and the Department Head of the Department of Industrial and Management Engineering, POSTECH. His research interests include vehicle routing problems, industrial optimization problems, generic simulation, healthcare optimization, and logistics. He was a Senior Member of the IISE. He is a member of INFORMS, KIIE, and KORMS. He received the Franz Edelman Finalist Award 2004 from INFORMS.