Toward delay-tolerant multiple-unmanned aerial vehicle scheduling system using Multi-strategy Coevolution algorithm

Khosiawan, Yohanes; Scherer, Sebastian; Nielsen, Izabela

Published in:
Advances in Mechanical Engineering

DOI (link to publication from Publisher):
10.1177/1687814018815235

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Research Article

Advances in Mechanical Engineering

Advances in Mechanical Engineering
2018, Vol. 10(12) 1–15
© The Author(s) 2018
DOI: 10.1177/1687814018815235
journals.sagepub.com/home/ade
SAGE

Toward delay-tolerant multiple-unmanned aerial vehicle scheduling system using Multi-strategy Coevolution algorithm

Yohanes Khosiawan[1], Sebastian Scherer[2] and Izabela Nielsen[1]

Abstract

Autonomous bridge inspection operations using unmanned aerial vehicles take multiple task assignments and constraints into account. To efficiently execute the operations, a schedule is required. Generating a cost optimum schedule of multiple-unmanned aerial vehicle operations is known to be Non-deterministic Polynomial-time (NP)-hard. This study approaches such a problem with heuristic-based algorithms to get a high-quality feasible solution in a short computation time. A constructive heuristic called Retractable Chain Task Assignment algorithm is presented to build an evaluable schedule from a task sequence. The task sequence representation is used during the search to perform seamless operations. Retractable Chain Task Assignment algorithm calculates and incorporates slack time to the schedule according to the properties of the task. The slack time acts as a cushion which makes the schedule delay-tolerant. This algorithm is incorporated with a metaheuristic algorithm called Multi-strategy Coevolution to search the solution space. The proposed algorithm is verified through numerical simulations, which take inputs from real flight test data. The obtained solutions are evaluated based on the makespan, battery consumption, computation time, and the robustness level of the schedules. The performance of Multi-strategy Coevolution is compared to Differential Evolution, Particle Swarm Optimization, and Differential Evolution–Fused Particle Swarm Optimization. The simulation results show that Multi-strategy Coevolution gives better objective values than the other algorithms.

Keywords

Unmanned aerial vehicle, scheduling, metaheuristic, optimization, delay-tolerant

Date received: 30 November 2017; accepted: 29 October 2018

Handling Editor: Bailing Tian

Introduction

Unmanned aerial vehicle (UAV) scheduling has been scrutinized by researchers in various application domains. Some researchers pursue green UAV operations, while some others seek to efficiently fulfill important tasks in an emergency situation.[1,2] The scheduling process involves a very large combinatorial space, a plethora of constraints and uncertainties—making the scheduling problem difficult to solve.[3] The reported works comprise not only UAV applications in outdoor environments but also the ones indoor. In indoor manufacturing environment, Khosiawan et al.[4] investigated the optimization of UAV operations for inspection and material handling. UAVs are assigned to

[1]Department of Materials and Production, Aalborg University, Aalborg, Denmark
[2]Carnegie Mellon University, Pittsburgh, PA, USA

Corresponding author:
Yohanes Khosiawan, Department of Materials and Production, Aalborg University, Fibigerstraede 16, 9220 Aalborg, North Jutland, Denmark.
Email: yok@mp.aau.dk

perform multiple tasks in the given environment. Nigam et al.[5] performed a study on parallel surveillance with multiple UAVs in an indoor test facility. The multiple task execution problem can be considered as a transportation problem (with zero payload or unlimited cargo capacity), where a UAV flies from one location to another to execute a task. Furthermore, a UAV executes one task after another (and possibly recharges its battery) in a planning horizon. In the transportation problem, Verderame et al.[6] address that the main challenges lie in the areas of fleet sizing, vehicle routing, and scheduling.[7,8]

This article focuses on the task scheduling system of bridge inspection operations using UAVs. It has the characteristic of both indoor and outdoor environments. While navigating beneath the bridge, the Global Positioning System (GPS) signal is not reliable since it is blocked by the deck.[9] The positioning system can utilize ultra-wideband, inertial measurement unit (IMU) or light detection and ranging (LiDAR). To enhance the accuracy and/or reliability, they can be combined and filtered based on the measurements' confidence levels. This technique brings flight uncertainty into play. Being outdoor, the existence of wind causes the real flight time to deviate from the scheduled one. Another constraint is the maximum flight time, by which the operational time is limited. The objectives of the scheduling system are to optimize the combination of the task value, makespan, and battery consumption. In other words, it maximizes the throughput and minimizes the cost of operation. Such objectives require a delay-tolerant scheduling system, which measures the impact of the uncertainty. It means that the schedule overall yields an optimized objective which relates to different considered measurements of uncertainty.

Regardless of the application domain, a scheduling system can consider uncertainties to yield a robust solution. Wu et al.[10] consider a scheduling problem where the release dates of the tasks are uncertain. The uncertainty is characterized by the estimated probability distribution of the arrival time. This assumption is highly dependent on the actuality of the observations. When the behavior of one of the involved elements in the operations is changed, the value is prone to differ. Li and Ierapetritou[11] present a scheduling system which considers uncertain processing time. In the problem formulation, the authors address the processing time uncertainty which is relevant to the addressed problem in this article. The uncertainty is characterized by an estimated variability of the processing time. This is realistic when the granularity of the task is made as atomic as possible. Hence, for that atomic process, a reasonable experiment-based variability can be considered in the scheduling.

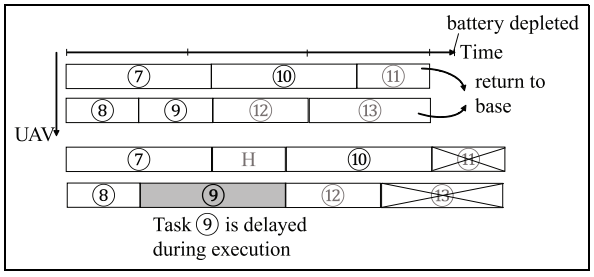

Figure 1. Inapplicable schedule due to execution uncertainty.

Figure 1 illustrates the impact of execution uncertainty on a schedule. The schedule instance corresponds to a plan of tasks to be executed within a given makespan MS_{max}. When the scheduling system optimistically assigns the tasks, the schedule becomes tight. A delayed completion time of a task (task ⑨) prevents the UAVs to return to base safely. Furthermore, task ⑩ needs to wait for the completion of task ⑨—given that task ⑨ is task ⑩'s predecessor. If the command dispatcher is intelligent, recharge tasks are carried out before tasks ⑩ and ⑫—with the drawback of MS_{max} being violated. The task assignments could have been arranged in a more realistic way which can cope up with the unknown known uncertainties.

Herroelen and Leus[12] brought the insight on reactive scheduling in their study. Reactive scheduling does not cope with uncertainty in creating the baseline schedule but revises the baseline schedule when an uncertain event occurs. In this article, the identified uncertainties are taken into account during the schedule generation—discouraging nervousness on the scheduling system. The uncertain flight time and task execution are patched with calculated slacks, which act as cushions. This means that the schedule allows the whole execution to finish at its earliest end time (when no uncertainty occurs), yet still considers a near optimum cost when uncertainty occurs. Due to this behavior, the respective algorithm is named Retractable Chain Task Assignment algorithm (RCTA).

Davenport et al.[13] have studied the usage of slack-based techniques for creating robust schedules. In the reported work, the temporal protection gives a slack toward the duration of the task. The amount of the temporal protection is the duration of uncertainties that are expected to occur during the execution of the task. The distinction of the presented RCTA is that there are temporal lower slack and upper slack, whose roles are different. The temporal lower slack is responsible for handling the delayed preceding or predecessor task, while the upper one is responsible for handling the current task's delay (see "Mathematical formulation and methodology" section for details). The earliest

Algorithm 1. Task Sequence Repair Algorithm.

```
   Input: task sequence (q)
   Output: feasible task sequence (q_f)
1: q_f ← null
2: while q is not empty do
3:     for task in q do
4:         for d in q_f do
5:             if task.predecessorList contains d then
6:                 remove d from task.predecessors
7:             if task.predecessorList is empty then
8:                 break
9:         if task.predecessorList is empty then
10:            add task into q_f
11:            remove task from q
12:            break
```

end time of a task is the earliest start time of the subsequent task, and the latest end time of a task takes the latest end time of the previous task into account. In this manner, the slack time of a task is sharable by all the subsequent tasks in a particular sequence, yet allowing the schedule to complete early (as if no slack is assigned at all) when no uncertainties occur. Consequently, a non-straightforward manner to evaluate the objective value of the schedule is needed and presented in the "Mathematical formulation and methodology" section.

Furthermore, researchers[14,15] have investigated real-time task scheduling system in dynamic and uncertain environments. In the reported works, the points of interest (in connection with the tasks) are moving during the mission execution. Pongpunwattana and Rysdyk[14] consider the possibility of multiple UAVs to cooperate on the same task, unlike Bertuccelli et al.[15] In both works, heuristic-based approaches are used. This is due to the nature of the problems being NP-hard. The most commonly used gradient-based techniques in the past can be either costly or even impossible to obtain the minima.[16] On the other hand, heuristic-based approaches, including metaheuristics, are suitable for exploring even the unlikely area in the search space to get a near optimum feasible solution in a reasonable amount of time.

In this study, RCTA is developed for constructing the schedule from a sequence. An individual in the population during the search is a sequence of tasks. This is due to the tractable manipulation of the sequence representation, which is required during the search. The sequence is yet transformable to the detailed schedule representation (e.g. with the timestamp, trajectory, and position of interest properties) by using RCTA. For the search, a metaheuristic algorithm called Multi-strategy Coevolution (MC) is proposed. Some researchers[17,18] investigate evolutionary algorithms which employ multiple strategies on subpopulations. On the other hand, MC clones the population to allow the individuals to experience each strategy simultaneously. The main population is called the elite group, where after a number of iterations will be replaced through a tournament selection of the other groups. This process iterates until the termination condition is met.

In addition, the UAV operations can be exposed to a constraint of maximum total makespan T_{max}. Depending on the sequence of execution of the task, the respective flight cost to go to the point of interest can be different. This constraint becomes an input for RCTA. RCTA assigns a task into the schedule when the total makespan is less than T_{max}. Otherwise, the task is discarded—it means that there is no UAV which can make it before T_{max}. During the search, MC maximizes the total value of the tasks in the schedule. When there is no T_{max}, MC minimizes the makespan. In both cases, the objective is followed by battery consumption minimization.

The main contributions of this study are listed as follows:

- Developed RCTA to construct a schedule from a sequence.

The constructed schedule incorporates slack time which acts like a cushion. Hence, the start time of a following task can be retracted as early as the end time of a preceding task. Yet, the latest end time takes into account a realistic slack time in connection with the task type and the required flight.

- Developed Multi-strategy Coevolution algorithm to perform the search.

This metaheuristic algorithm allows each individual in the population to be exposed to each strategy at the same state. The elite group is cloned according to the number of search strategies, and each clone group is exposed to a search strategy. The clone groups are linked to the elite one through a tournament selection which is done every couple of iterations. The result of the tournament selection replaces the elite group. This process iterates until the termination condition is met.

The remainder of the article is organized as follows. The description of the problem is addressed in the "Problem definition" section. The proposed methodology: RCTA and MC are explained in detail in the "Mathematical formulation and methodology" section. The results of numerical experiments and analysis are presented in the "Numerical simulations" section. The concluding remarks of the study are then presented in the "Conclusion" section.

Problem definition

This work investigates the scheduling of multiple-UAV operations in a bridge inspection, which considers execution delays due to flight uncertainty. There are multiple tasks representing multiple inspection areas throughout the bridge. UAV performs the given tasks in accordance with the environment specifications.[4] In a given time horizon, the corresponding tasks are planned as a schedule, which will be executed afterward. A UAV executes a task at a time, and the task can only be executed once. Since the task is executed at a particular geographical location, the position data become a relevant factor to look at. The values in the data are influenced by the flight uncertainty factors such as the mechanical condition of the UAV, the environmental condition (e.g. wind), and the measurement data from the positioning system during the task execution. In Figure 2(a) and (b), the planned flight path andits execution are depicted through waypoints (o) and paths (-). The flight uncertainty causes the execution to deviate from the planned flight. The deviation distances for all planned waypoints are measured. Through a one-sided t-test (with the null hypothesis of the mean distance equals 0, against the alternative where the mean is greater than that), they are statistically significantly deviating from the planned ones—with p value $= 5.8585e-16$, $\alpha = 0.05$.

Upon the observation of the data, the flight approaching and leaving a waypoint tend to experience delays. This corresponds to the aforementioned flight path deviation during the execution. While approaching a target waypoint, the UAV slows down and checks whether it has arrived at the desired position. When the UAV is within <1 m radius, it considers that it is at that particular location. This is related to the dimension of the UAV which can be represented at least as a sphere with a diameter of 0.5 m. As the required confidence gets higher, getting an accurate actual position measurement can take longer time. Furthermore, to fly toward the subsequent waypoint, the UAV might need to adjust its orientation (in respect to the Euler angles). Thus, the magnitude of the waypoints contributes to the scale of the delayed flight execution time.

To deal with such an operational environment, the schedule shall be delay-tolerant. This means that the comprised tasks in the schedule are neither under- nor over-assigned—in respect to the involved constraints (e.g. battery consumption, maximum makespan). This is measured through the robustness of the schedule in the proposed approach.

This study addresses the problem of multiple-UAV scheduling system with a delay-tolerant awareness. Multiple inspection tasks are performed by multiple UAVs, whose execution time uncertainties can delay

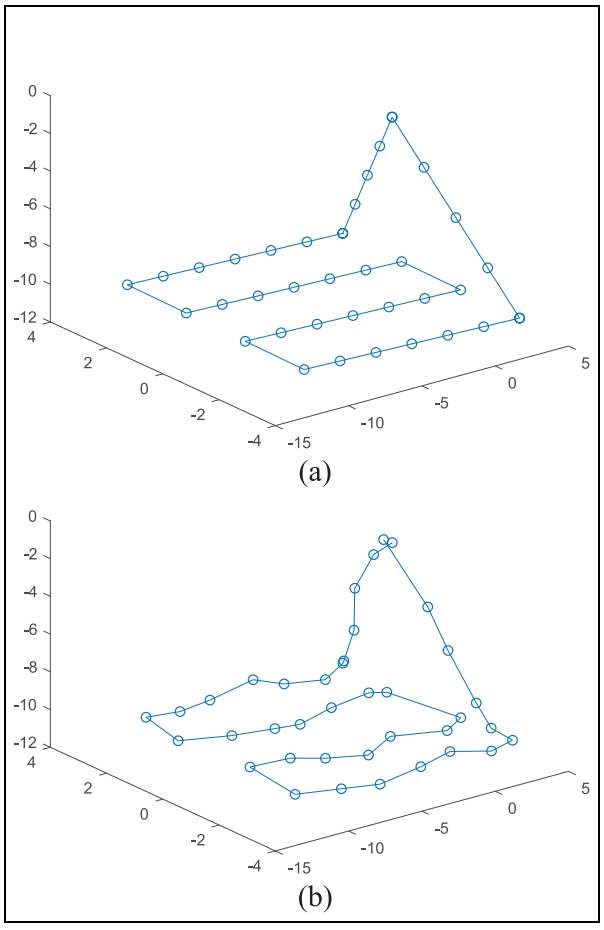

Figure 2. (a) Planned flight path and (b) its execution uncertainty.

the scheduled completion time. The tasks can have precedence relationship, and the predictive execution time is determined. A fixed-time schedule of the tasks introduces high nervousness, in which flight conflicts arise and a high operational cost is incurred. An approach which is based on slack time is proposed in this study. Consequently, a task schedule is encapsulated by an earliest start time and a latest end time—other properties are explained in detail in the "Mathematical formulation and methodology" section. A corresponding schedule instance is depicted in Figure 3. The overlapping timespans of tasks assigned to a UAV (e.g. timespans 0–48 and 44–55 on UAV 101) are related to the proposed method which is explained in the "Mathematical formulation and methodology" section. Upon the schedule completion, the remaining battery power of a UAV is ensured to be sufficient for returning back to recharge station. For conciseness, the final trip back to recharge station is not explicitly included in the schedule.

In coherence with the aforementioned goal, the objectives of minimizing makespan and battery

```
Schedule: {
        101=[to_OR-L6(0-115), 9(107-115), to_BD-
M6(107-173), 4(160-180)],
        102=[to_BD-M3(0-60), 3(50-67), to_BD-
UR3(55-81), 5(61-88), to_BD-UL3(66-104), 1(78-
111), to_BD-LL3(83-118), 2(87-125), to_OR-
U3(92-166), 10(129-184)],
        103=[to_BD-LR3(0-53), 6(45-60), to_OL-
L3(50-99), 7(80-99), to_OL-U6(80-161), 8(138-
179)]
}
```

Figure 3. Illustration of a schedule instance.

Table 1. Task type.

Task type	Description	Task value	Processing time (s)
1	Large-scale inspection	5	5
2	Detailed inspection	10	15
3	Hands-on inspection	20	25

consumption are pursued through schedule optimization. The primary objective is to minimize the makespan of schedule δ (denoted by $\mathbb{MS}(\delta)$), followed by minimizing the battery consumption of schedule δ (denoted by $\mathbb{B}(\delta)$)—refer equations (1) and (2), respectively

$$Minimize \quad \mathbb{MS}(\delta) \tag{1}$$

$$Minimize \quad \mathbb{B}(\delta) \tag{2}$$

Correspondingly, the tasks have task values, whose total is maximized. The task value correlates to the proportion of a task in completing the whole bridge inspection. The more in depth the (inspection) task type, the higher the task value. Furthermore, the processing time increases as the task type gets more in depth. Task value and processing time of different task types are depicted in Table 1. The types of the task include large-scale (e.g. on the deck surface), detailed (e.g. on the hanger), and hands-on inspections (e.g. on the joint of the hanger and the deck, and the joint of the hanger and the arch or the foundation). The regions of interest on the bridge are illustrated in Figure 4. A large-scale inspection can be done through image processing,[19] a detailed one can be with ultrasonic,[20] and a hands-on one can be a coordinated combination of both.

Moreover, a task might have predecessor tasks. A predecessor task is a task which is not necessarily assigned right before a particular task, but its completion must be prior to the commencement of the particular task. It differs from a preceding task, which is assigned right before a particular task in a UAV's schedule.

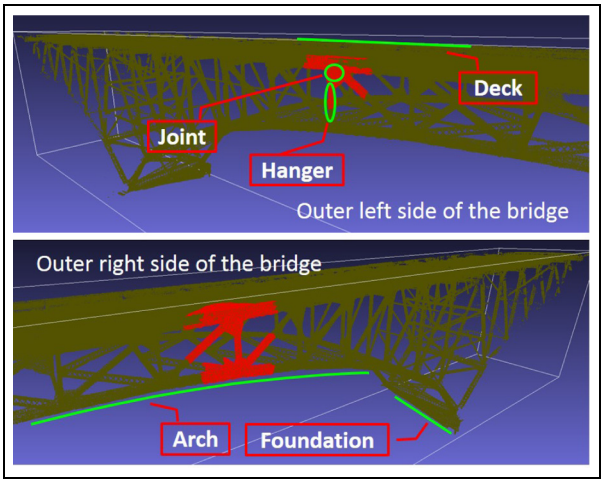

Figure 4. Regions of interest on the bridge.

In addition, a subproblem with a maximum makespan limitation is addressed. When the maximum makespan is limited, the primary objective is to maximize the total task value (equation (3)), followed by minimizing the battery consumption (equation (2)). The detailed mathematical formulation is presented side by side with the proposed methodology in detail in the "Mathematical formulation and methodology" section

$$Minimize \quad \mathbb{V}(\delta) \tag{3}$$

The backbone of the problem addressed in this article can be considered as traveling salesman optimization problem. This problem is known to be NP-hard,[21] where mathematical approaches require a significantly increased amount of time as the problem grows.[22] Hence, the mathematical approaches are only suitable for small-scale problems in practice.

The case which is used in the "Numerical simulations" section is depicted in Figure 5(a). The forward part of the bridge is shown with the available waypoints and paths; see Figure 5(b) for more emphasis on waypoints and paths. Figure 5(c) depicts an isometric view which mostly covers the view beneath the arch of the bridge, illustrating the labeled waypoints and paths which are used during the scheduling. A corresponding instance of task data is depicted in Table 2. The aforementioned schedule instance in Figure 3 comprises the tasks in Table 2. In this study, the simulation focuses on the flight uncertainty, while the task execution uncertainty is addressed in the proposed method. Hence, the scope of this work serves as a progress

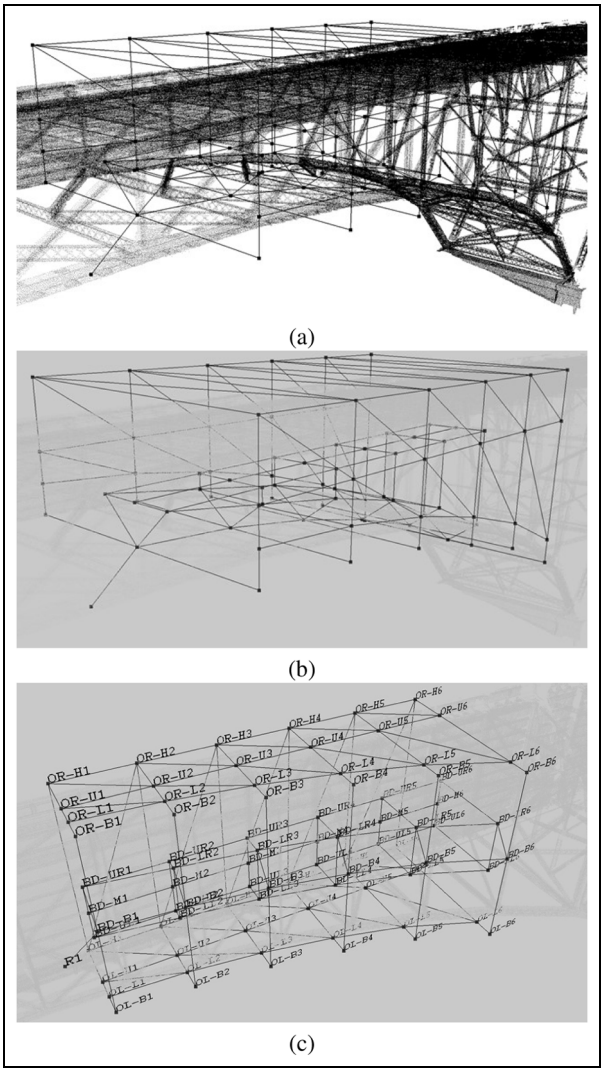

Figure 5. Forward part of a bridge.
Note: Labels in Figure 5c mark the positions and waypoints in the UAV operations. Positions in Table 2 correspond to these labels.

Table 2. Task data.

Task ID	Position	Task type	Predecessors (Task IDs)
1	BD-UL3	1	–
2	BD-LL3	1	–
3	BD-M3	1	–
4	BD-M6	1	–
5	BD-UR3	1	–
6	BD-LR3	2	–
7	OL-L3	3	–
8	OL-U6	2	7
9	OR-L6	3	–
10	OR-U3	2	9

toward the delay-tolerant multiple-UAV scheduling system.

Mathematical formulation and methodology

RCTA

In the UAV operations, there are uncertain events which can affect the schedule execution. This leads to the deviation of the real flight time from the scheduled one. The motivation of generating the schedule is to optimize the combination of total task value, makespan, and battery consumption. When a schedule which considers no uncertainty is executed under an environment with uncertainties, the purpose of the schedule becomes undermined. Hence, the proposed methodology seeks a delay-tolerant schedule, which overall yields an optimized objective which relates to different considered measurements of uncertainty.

RCTA allows a flexible amount of time between two adjacent tasks in a schedule. A temporal lower slack is used in consideration of the preceding and predecessor tasks. The earliest start time of task i assigned to UAV u, denoted by ξ_{ui}, equals the latest time among the earliest end time of the preceding task $\zeta_{u,i-1}$ and the predecessor tasks $\zeta(\phi_{ui})$ (see equation (4))

$$\xi_{ui} = \max\left(\zeta_{u,i-1}, \max_{\iota \in \phi_{u,i}}(\zeta_\iota)\right) \tag{4}$$

The latest start time of task i executed by UAV u, denoted by α_{ui}, equals the latest time among the latest end time of the preceding task, denoted by $\omega_{u,i-1}$, and the latest end time of the predecessor tasks ϕ_{ui}, denoted by $\omega(\phi_{ui})$ (see equation (5))

$$\alpha_{ui} = \max\left(\omega_{u,i-1}, \max_{\iota \in \phi_{u,i}}(\omega_\iota)\right) \tag{5}$$

When no delay occurs, the schedule hence can be executed as if no slack is assigned—yet the slack time can be used by every subsequent task in a particular sequence. When the latter part of the schedule is prone to more delay (e.g. by the delay that occurred in the beginning of the schedule), the unused former slack provides more tolerance.

Equation (6) states the earliest end time of task i assigned to UAV u, denoted by ζ_{ui}, equals the earliest start time ξ_{ui} plus the duration of the task $d_{0_{ui}}$

$$\zeta_{ui} = \xi_{ui} + d_{0_{ui}} \tag{6}$$

Equations (7) and (8) show the latest end time of task i assigned to UAV u, denoted by ω_{ui}, equals the latest start time α_{ui} plus the duration of the task and the respective slack d_{ui}. This is where the upper slack is used in consideration of the current task's delay. Depending on the type of the task, the respective task slack time is

contributed by $c_{\lambda_{ui}}$. When it is a flight action, the slack time is contributed by $w_{\lambda_{ui}}$

$$\omega_{ui} = \alpha_{ui} + d_{ui} \tag{7}$$

$$d_{ui} = d_{0_{ui}} + c_{\lambda_{ui}} + w_{\lambda_{ui}} \tag{8}$$

A task has a specialization as an action; it can be a flight, wait-on-ground, hover, or battery replacement action. A flight action has uncertainty in its execution time. On the other hand, hover, wait-on-ground, and battery replacement actions have no execution uncertainty, setting the slack time to zero. In connection to this classification, the slack time of a task can be $c_{\lambda_{ui}}$ or $w_{\lambda_{ui}}$ as depicted in equations (9) and (10). For a task, the slack is predetermined based on the type of the task t_{ui}, that is, large-scale, detailed, and hands-on inspections. For a flight action, it is based on the number of traversed waypoints η and a variable flight slack λ_f. A priori observations show that the UAV experiences more execution uncertainty when it tries to fly through multiple waypoints than just cruising on a straight line between two waypoints. This phenomenon drives the granularity of the flight delay time to be in respect to the number of the traversed waypoints (η). The flight delay time in regard to a waypoint determines the value of the variable slack λ_f

$$c_{\lambda_{ui}} = \lambda_t(t_{ui}) \tag{9}$$

$$w_{\lambda_{ui}} = (\eta - 1)^* \lambda_f \tag{10}$$

The value of λ_f is not constant; it follows a normal distribution model based on the real test flight data. These data belong to the flight of a UAV following a plan depicted in Figure 2(a). The flight data are recorded in the form of Robot Operating System (ROS) messages, whose properties include position and timestamp.[23] The message is broadcasted at the frequency of 50 Hz—see Figure 6(a) for the whole data (notice that Figure 2(b) is constructed by taking the closest corresponding waypoints from the data in Figure 6(a) in connection with the planned ones in Figure 2(a)). The data are then sampled in connection with the planned waypoints. At each planned waypoint, messages whose positions are within 1 m are collected. This is related to the reference system of the UAV which considers a UAV to be at a particular position when it is <1 m in distance. Then, the respective timestamps are formed into groups based on the corresponding planned waypoint. The biggest difference of timestamp in each group is calculated, and the respective data from all groups are estimated to follow a normal distribution, where $\hat{\mu} = 3.83$ and $\hat{\sigma} = 3.05$. These values indicate a reasonable flight uncertainty that incurs when the UAV approaches a waypoint and leaves it to go to the next one. The sampling is illustrated in Figure 6(b), in which (o) and (-) represent waypoint and path, respectively.

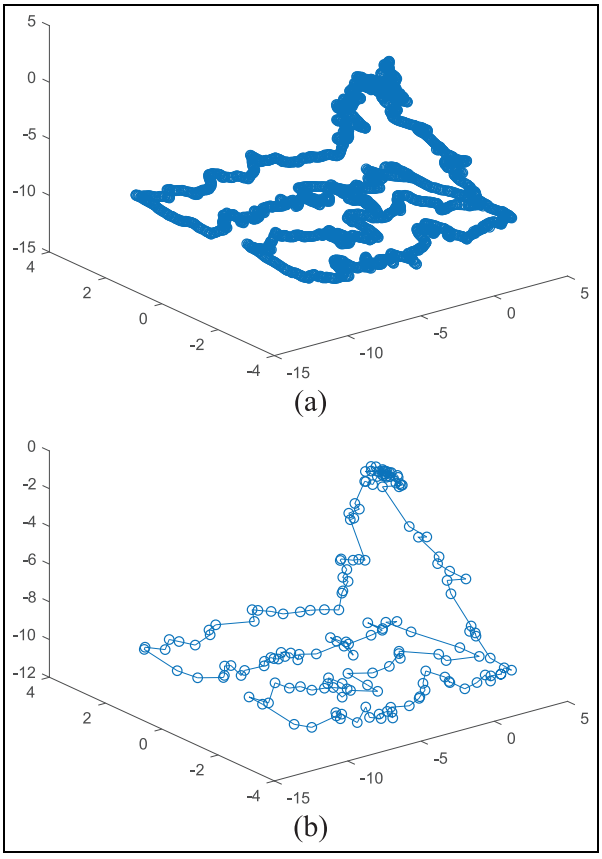

Figure 6. (a) Whole and (b) sampled ROS messages around the planned waypoints.

Furthermore, the UAV velocity is calculated and modeled based on previously obtained flight data. The flight data comprise the positions and timestamps of the UAV while operating at a constant velocity. A normal distribution is fit to the data, where $\hat{\mu} = 0.52$ and $\hat{\sigma} = 0.24$ are estimated. This is used in the numerical simulation to imitate the UAV flight performance. In the simulation, the velocity can be lower than average for an entire flight path—not just when it is approaching a waypoint. In this manner, the simulation puts the generated schedule in a harsher operational environment than the real one.

Equations (11) to (13) depict the battery level calculation. The battery consumption of task i which is executed by UAV u, denoted by b_{ui}, equals the difference between the latest end time of the task ω_{ui} and the preceding one $\omega_{u,i-1}$. It relates to the characteristic of the temporal lower slack and the upper slack which are incorporated into the schedule as depicted in Figure 7. This applies to all tasks except battery replacement and wait-on-ground actions, where $b_{ui} = 0$. Respectively, the battery level of UAV u after executing task i, denoted by B_{ui}, is decreased by b_{ui} when it is a task

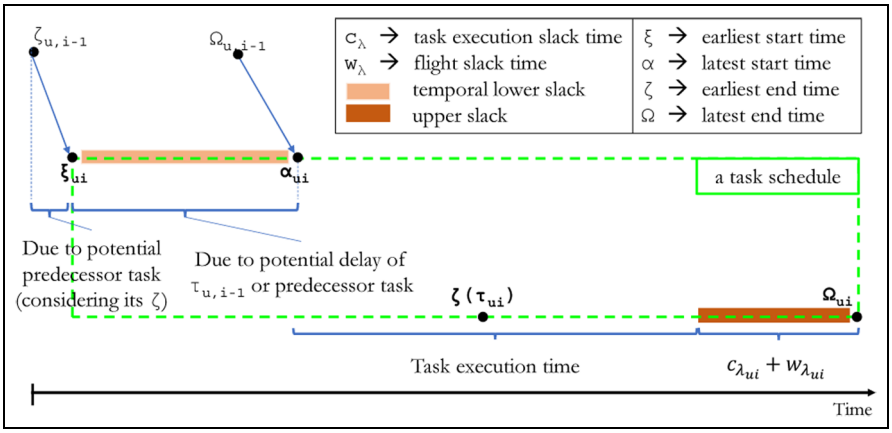

Figure 7. Structural representation of a task schedule.

execution, flight or hover action. In the case of battery replacement action, the battery level of the UAV is set to its full capacity F. Otherwise, the UAV sits on the ground, in which the battery consumption is assumed as 0. The battery replacement is scheduled before task i if there is not enough battery level on the UAV to go to the nearest recharge station after the task execution. In this manner, the final trip back to recharge station is accounted for in the scheduling. Letting j be the flight action from the position of task i to the nearest recharge station, the battery replacement triggering condition is depicted in equation (14)

$$b_{ui} = \omega_{ui} - \omega_{u,i-1} \tag{11}$$

$$B_{ui} = B_{u,i-1} - b_{ui} \tag{12}$$

$$B_{ui} = F \tag{13}$$

$$B_{ui} < b_{uj} \tag{14}$$

In the case where maximum makespan limitation MS_{max} exists, the insertion of a task is dropped when the yielded schedule makespan $MS(\delta)$ exceeds the defined limit. Otherwise, MS_{max} equals ∞ indicates the scheduling case without maximum makespan limitation. A flowchart of RCTA is depicted in Figure 8.

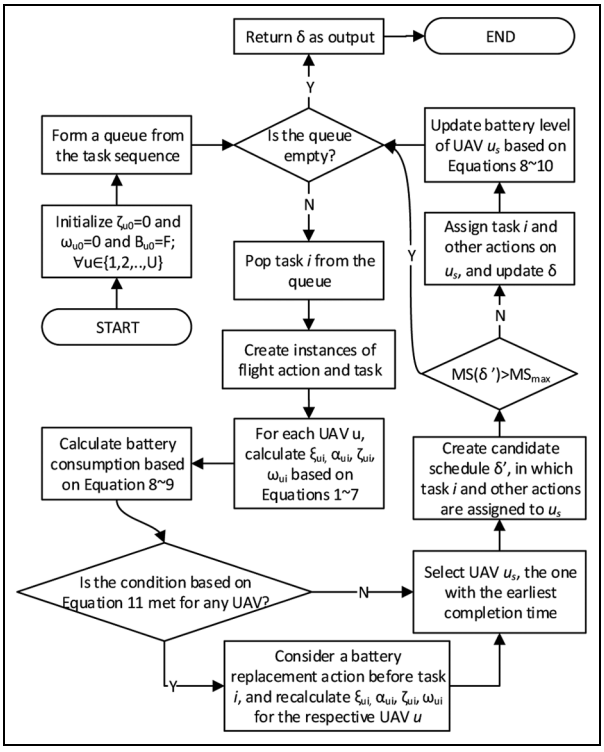

Figure 8. Flowchart of Retractable Chain Task Assignment algorithm.

MC

To search the solution space, a metaheuristic algorithm called Multi-strategy Coevolution algorithm is proposed. In principle, it utilizes linked clone groups which run different strategies during the search. The fitness evaluation of a solution candidate during the search is based on the defined objective(s).

The objective function of minimizing the makespan of schedule δ, denoted by $\mathbb{MS}(\delta)$, is depicted in equation (17). $\mathbb{I}_u$ is the number of tasks in a sequence which is assigned to UAV $u \in \mathbb{U} = \{1, 2, \ldots, U\}$, and $slack_{ui}$ equals the slack time $c_{\lambda_{ui}}$ or $w_{\lambda_{ui}}$, depending on whether it is a flight action or a task (note that an action is a specialization of task). The naive makespan of UAV u, denoted by MS_u, is calculated according to equation (15). Correspondingly, the naive makespan of the entire schedule δ, denoted by $MS(\delta)$, is calculated according to equation (16)

$$MS_u = \omega_{ui} \quad i = \mathbb{I}_u \tag{15}$$

$$MS(\delta) = \max_{u \in \mathbb{U}} (MS_u) \tag{16}$$

The usage of the natural logarithm is to avoid excessive slack time. This is because the value which is

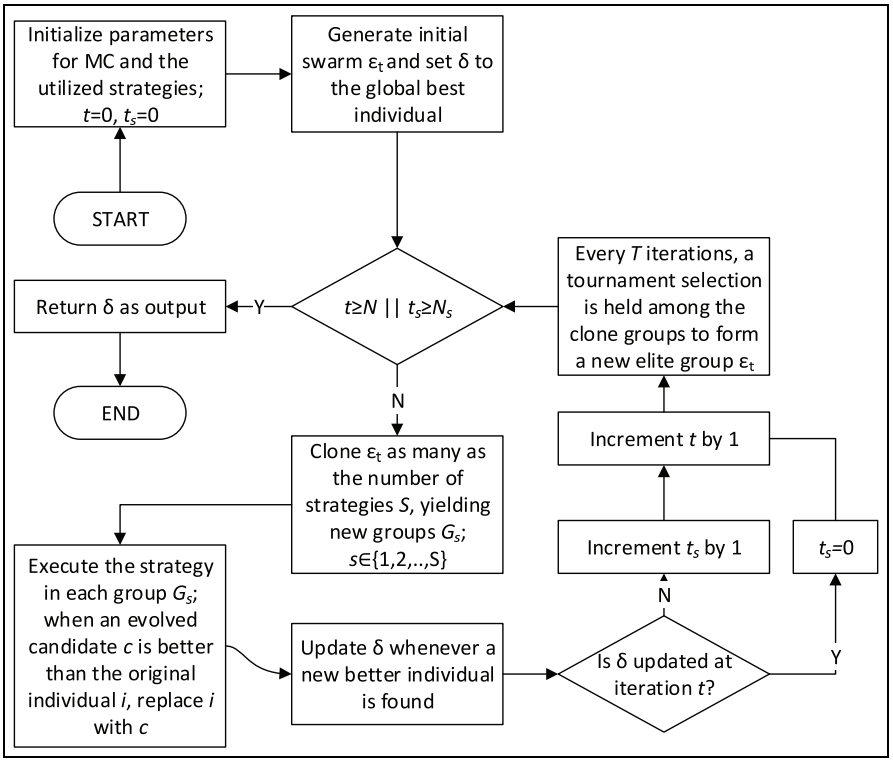

Figure 9. Flowchart of Multi-strategy Coevolution algorithm.

subtracted from MS_u becomes normalized (not excessive). A bit longer slack can be accepted, but as it gets longer, more battery replacements are required. It leads to a longer makespan, and, respectively, lower chance of the schedule to be selected. The usage of $(\mathbb{I}_u - i) + 1$ allows a bigger slack time for tasks in the latter parts of the schedule. This is due to their higher likelihood of getting delayed, which is translated from the interference on the former tasks

$$\mathbb{MS}(\delta) = \min\left(\max_{u\in\mathbb{U}}\left[MS_u - \sum_{i=1}^{\mathbb{I}_u}\frac{\ln(slack_{ui})}{(\mathbb{I}_u - i) + 1}\right]\right) \quad (17)$$

When the maximum makespan limitation exists, the objective is to maximize the total task value $\mathbb{V}(\delta)$, whose function is depicted in equation (18)

$$\mathbb{V}(\delta) = \sum_{u=1}^{U}\sum_{i=1}^{\mathbb{I}_u} v_{ui} \quad (18)$$

The aforementioned objective functions are primary for the, respectively, mentioned cases (i.e. without and with maximum makespan limitation). In both cases, the primary objective function is followed by a secondary objective of minimizing the battery consumption $\mathbb{B}(\delta)$, whose function is depicted in equation (19). When multiple solutions have equal primary objective value, the secondary one breaks the tie. The battery consumption is quantified in terms of its lifetime. In this pilot study, the rate is assumed to be constant over time (i.e. when the UAV flies toward a position, hovers, or executes a task)

$$\mathbb{B}(\delta) = \sum_{u=1}^{U}\sum_{i=1}^{\mathbb{I}_u} b_{ui} \quad (19)$$

The detailed procedure of MC is explained as follows, and the respective flowchart is depicted in Figure 9.

Step 1. Initialize the parameters of the utilized strategies. For MC itself, initialize the size of population P, period of the tournament selection T, the maximum number of iterations N, and the maximum number of stagnant iterations N_s (no improvement seen). In this study, P equals 40, T equals 10, N equals 40, and N_s equals 10. For the crossover operations, the crossover probability CR equals 0.5. These values are selected based on a priori simulations.

Step 2. Generate the initial swarm based on the priority rules which are intuitively believed to produce good starting points. When the number of priority rules is less than P, the remaining individuals are generated by mutating the existing ones. From this point forward, the initial swarm is referred as the

elite group ϵ_t, where the iteration index t equals 0. In addition, the counter of stagnant iterations t_s equals 0.

Step 3. If $t \geqslant N$ or $t_s \geqslant N_s$ has been met, end the search and take the best solution δ as the output. Otherwise, go to step 4.

Step 4. Clone ϵ_t as many as the number of the utilized strategies S. In this study, two strategies are used: (1) crossover between an individual in the group and a random one in ϵ_t and (2) crossover between an individual in the group and the best solution. The crossover point is random at any given time. When a crossover is performed, a simple repair mechanism which follows Algorithm 1 is performed—this serves as the mutation experience. Correspondingly, together with ϵ_t, there are $S + 1$ groups in total. The execution of a strategy in a group allows interactions with the elite group, but only limited to read access. Furthermore, δ is shared by all the groups throughout the entire search.

Step 5. For each iteration, each clone group will execute its strategy on a certain individual to produce a new candidate c. If c is better, replace the individual with c. When c is better than the current δ, set δ to c. This operation can be performed on all individuals or partly, according to the defined strategy.

Step 6. In the end of an iteration, increase t by 1. If there is no improvement on the best solution in this iteration, increase t_s by 1.

Step 7. For every T iterations, a tournament selection is held among the clone groups. The best P individuals are selected to form a new elite group ϵ_{t+1}, discarding the old one ϵ_t. Go to Step 3.

Furthermore, the robustness level of the schedule is calculated in a straightforward way. For every task (i.e. action or task) in the schedule, the scheduled time is compared against the one from the simulation. If the simulated execution time is contained inside the scheduled one, it is put into the set of conformed tasks T_c, otherwise it is put into the violated set T_v. Correspondingly, the robustness level $\mathbb{R}(\%)$ is depicted in equation (20)

$$\mathbb{R} = \frac{|T_c|}{|T_c| + |T_v|} * 100 \tag{20}$$

Numerical simulations

In the numerical simulations, scenarios of inspection on a site depicted in Figure 5 with 2 up to 10 UAVs are carried out. The number of tasks N_t is set to 52 in the data of every scenario. As mentioned in the "Problem definition" section, this study inclines toward the flight uncertainty investigation. Consequently, the task execution uncertainty is not included in the simulation. The simulations include comparisons of the proposed method MC with Differential Evolution (DE), Particle Swarm Optimization (PSO), and Differential Evolution–Fused Particle Swarm Optimization (DEFPSO—introduced by Khosiawan et al.[24]). DE and PSO are prominent metaheuristic algorithms among others, which have been used in various optimization problems.[16,25] To have a fair comparison, all metaheuristic algorithms (DE, DEFPSO, MC, and PSO) are given the same P, N, and N_s. The crossover operation in DEFPSO and MC is a one-point crossover, with a random crossover point following a uniform distribution ($a = 0$, $b = N_t - 1$). The parameter values for each algorithm are listed as follows.

- DE

$F = 0.8$ (weighting factor which controls mutation) and $CR = 0.5$ (crossover control parameter)

- DEFPSO

$F = 0.5$ (F acts similar to c_2 in PSO), $CR = 0.5$

- MC

$T = 10$ and $CR = 0.5$

- PSO

$c_1 = 1$ (cognitive learning coefficient) and $c_2 = 2$ (social learning coefficient), while u_1 and u_2 are randomly (following a uniform distribution) set in the range of [0, 0.5].

- DE, PSO, DEFPSO, and MC

$P = 40$ (size of population), $N = 40$ (maximum number of iterations), and $N_s = 10$ (maximum number of stagnant iterations).

In the following figures (Figures 10–14), the measurement of the schedule properties is depicted in boxplots and fitted lines. A boxplot depicts the distribution of data based on the five-number summary (minimum, first quartile, median, third quartile, and maximum). A fitted line depicts a local polynomial regression fitting[26] with *span* = 0.75 and *degree* = 2.

At a glance, the makespan of the schedule is obviously decreased when more UAVs are used (see Figure 10). However, this observation needs a further investigation on its robustness. Through the simulations, more waiting time due to the high path occupations are found. More delays are exposed and the robustness of the schedule is decreased. On the other hand, this condition has a compensation when the tasks are more distributed to the higher number of

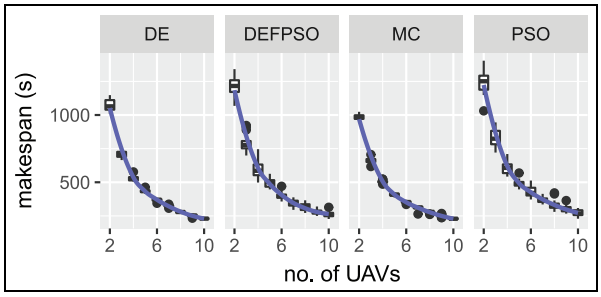

Figure 10. Makespan of the schedule based on the simulations.

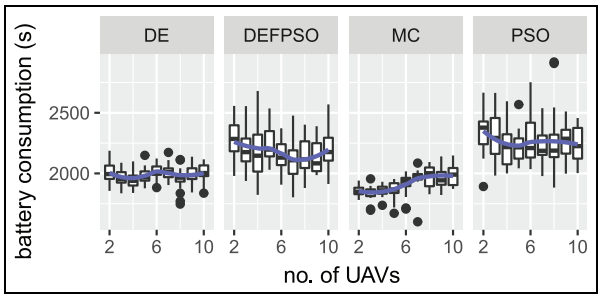

Figure 11. Battery consumption of the schedule based on the simulations.

UAVs. Distinctly assigned areas are likely to produce optimized makespan, the sequence assigned to each UAV is not long, and hence the battery consumption can be decreased. In a different setup, where more alternative paths are available, the robustness deficiency can be alleviated at the cost of longer makespan and battery consumption.

The effects of the occupied path and the less assigned tasks (to each UAV) keep pulling each other. Figure 11 depicts this phenomenon. In the beginning, the battery consumption is decreased as the number of UAVs begins to increase. As it gets higher (the exact number differs from one algorithm to another), the overhead cost of departing and going back from and to the base is increased. Since the paths are clogged, a higher amount of waiting time is exposed—which can create more delay and decrease the robustness. In opposition to that, the more distributed tasks yield less tasks for each UAV to execute, and the battery consumption for that is less. As a result, a sin-like pattern can be seen in the depicted boxplots.

The robustness degrades as the limited number of paths (especially when they depart from one single station/base) is clogged by numerous UAVs. Figure 12(a) depicts the consequence of the delayed completion time caused by the frequently occupied paths. It makes the execution time of the planned tasks deviate from the schedule. This degradation can be reduced by tracking the path occupation during the schedule generation. This brings a drawback of an increased computation time, though. In addition, the respective schedule potentially has more hover actions, whose total battery consumption is not attractive. Moreover, the diminishing robustness in the presented result helps to indicate the effective number of UAVs to be used.

To present a further investigation on such a phenomenon, the robustness levels of schedules considering a large number of competitive alternative paths are presented in Figure 12(b). It means that an alternative path exists when the shortest one is occupied, and their costs are equal. Another factor which degrades the robustness level as the number of UAVs increases is the

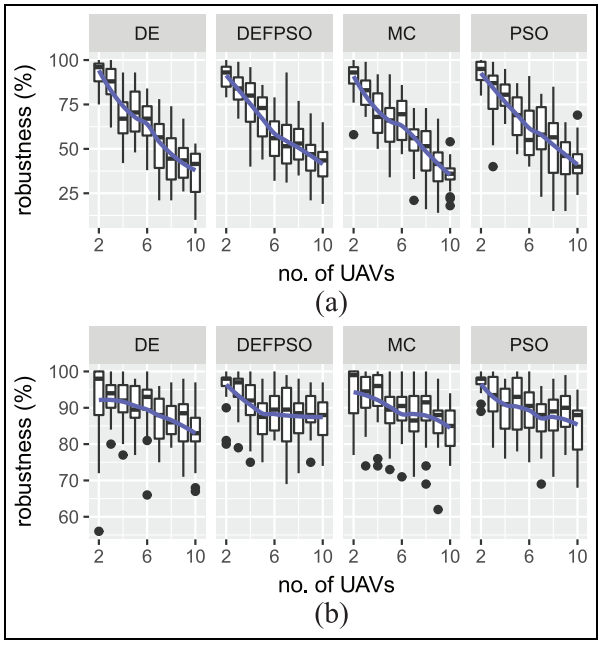

Figure 12. Robustness level of the schedule based on simulations with (a) limited and (b) large number of competitive alternative paths.

diminishing cumulative buffer effect. When the tasks are highly distributed, each UAV has less assigned tasks. This leads to less unused slack time which can be contributed from the former to the latter part of the schedule. Notice that the UAV engine failure is not included in the simulation in this study.

In connection with the performance of the proposed metaheuristic algorithm, schedules from MC have less makespan and battery consumption than DE, PSO, and DEFPSO. Furthermore, the lower bound of the robustness is slightly better than others. The computation time of the schedule generation with MC is reasonable, even though it is not the fastest one (see Figure 13). This can be seen as the trade-off with the better other properties of the generated schedule.

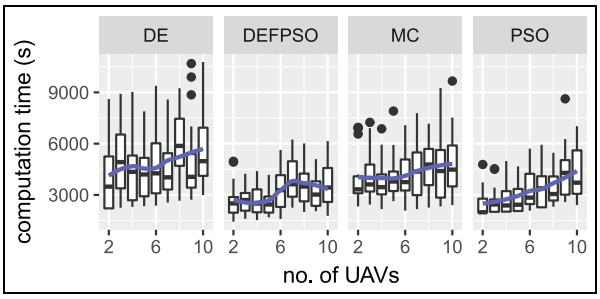

Figure 13. Computation time of the schedule generation based on the simulations.

Correspondingly, performing the operations with 2 UAVs looks reasonably good in this case study.

Numerical simulations for operations with maximum makespan limitation are also conducted. The results are depicted in Figure 14 and the respective analysis is presented as follows. Let MS_{max} be 600 s (10 min), one can consider the utilization of 3 to 4 UAVs as a viable option. MC gives better objective values, which are task value and battery consumption in this case, than the other algorithms. Specifically, the task value can be highly obtained when 3 or 4 UAVs are used, and all tasks are shown to be covered from that point onward. Consequently, the battery consumption goes relatively stagnant when 4 and more UAVs are utilized. An observation on the computation time of MC shows that it reaches the peak at 6 to 7 UAVs. This shows the characteristic of metaheuristic where it might get attracted to a local optimum. When such a condition holds for more than N_s iterations, it caused a few simulation runs of 10 UAVs to end immediately. Based on Table 3, the average computation time of MC on 7 and 10 UAVs are 4.78 and 4.34 s. It indicates that the peak is quite marginal.

The schedule makespan is decreased as the tasks are more distributed among the UAVs. The corresponding computation time scales, in regard to the solution space which gets bigger following the higher number of UAVs. As depicted in the previous case ($MS_{max} = \infty$), the robustness level degrades as more UAVs are involved. In both cases, the effectiveness of the proposed method is aligned to the balance between the number of the deployed UAVs and the map. The amount of 2 to 4 UAVs is considered appropriate to enhance the efficiency while reducing the overhead and waiting costs. This also gives a mean robustness level of $\approx 75\%$ and above. When the large number of competitive alternative paths is used, the robustness level can be maintained better. Figure 15 depicts the mean robustness level of >90% when 2 to 4 UAVs are used. This is consistent with Figure 12(b).

Table 3 lists the mean makespan ($\overline{\mathbb{MS}}$), mean battery consumption ($\overline{\mathbb{B}}$), mean computation time ($\bar{\tau}$), and

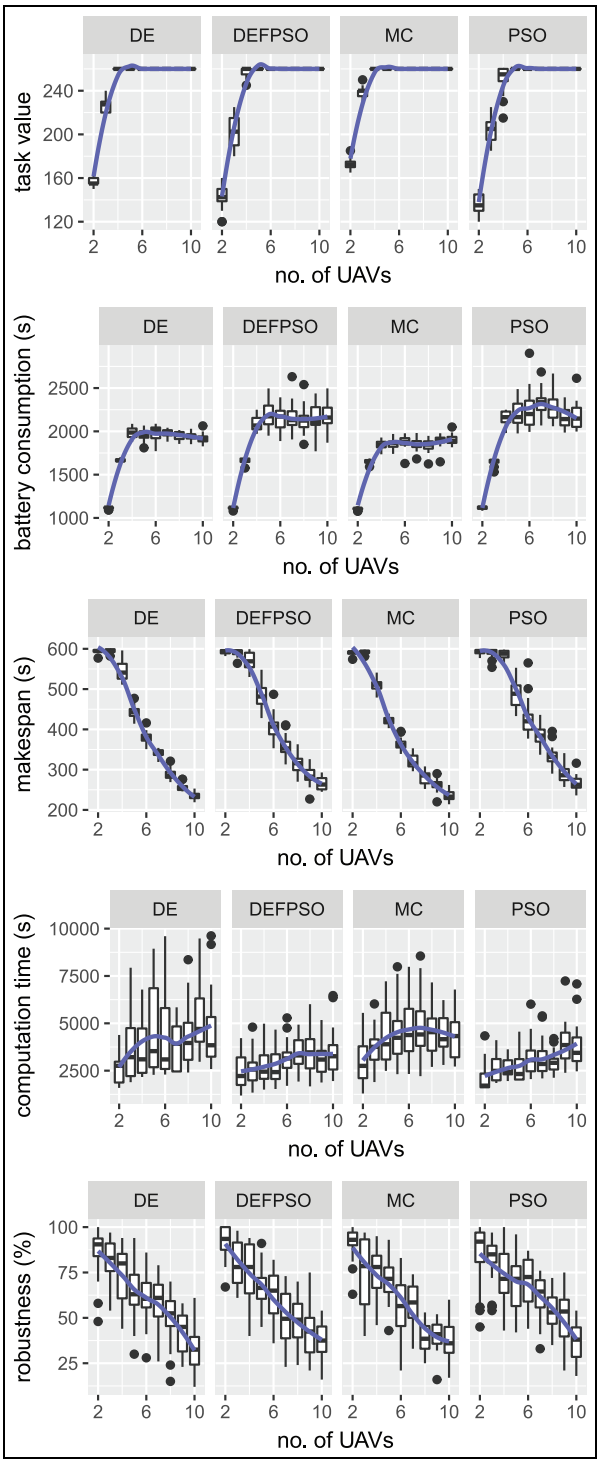

Figure 14. Simulations with maximum makespan limitation.

mean robustness level ($\overline{\mathbb{R}}$) for different numbers of UAVs (U) and algorithms (A). The mean task value ($\overline{\mathbb{V}}$) is shown when $MS_{max} = 600$ s, while it is omitted ($\mathbb{MS}$ always equals 260) when $MS_{max} = \infty$. The best values among the compared algorithms are marked with asterisks, and the ones yielded by MC are shaded (when the values from all algorithms are equal, no asterisk is

Table 3. Means of the numerical simulation results.

No.	U	A	$MS_{max} = \infty$				$MS_{max} = 600$				
			$\overline{\mathbb{MS}}$(s)	$\overline{\mathbb{B}}$	$\overline{\tau}$(s)	$\overline{\mathbb{R}}$(%)	$\overline{\mathbb{V}}$	$\overline{\mathbb{B}}$	$\overline{\mathbb{MS}}$(s)	$\overline{\tau}$(s)	$\overline{\mathbb{R}}$(%)
1	2	DE	1071.8	2007.3	3.86	93.3	157.5	1116.4	594.2	2.62	85.8
2	2	DEFPSO	1209.5	2274.55	2.65	91.45	143	1115.05	592.45	2.44	91.7*
3	2	MC	986.3*	1854.65*	3.96	90	172.5*	1105.35*	590.45*	3.05	91.05
4	2	PSO	1241.55	2338.7	2.48*	93.6*	136.5	1117.1	591.85	2.11*	84.4
5	3	DE	704.65	1968.4	5.12	86.5*	226	1664.4	594	3.72	82.1*
6	3	DEFPSO	792.95	2205.75	2.67	83.25	203.75	1659.6	590.5	2.58*	79.8
7	3	MC	660.35*	1838.15*	4.19	82.45	238.5*	1648.7	591.85	3.67	74.95
8	3	PSO	831.25	2310.15	2.60*	81.4	204.25	1647.75*	588.9*	2.66	82.1*
9	4	DE	530.7	1946.1	4.42	68	260*	1982.85	543.75	3.67	74.5
10	4	DEFPSO	594.75	2169.1	2.66*	75.1	257	2095.65	570.15	2.78	75.85*
11	4	MC	504.25*	1858.3*	3.84	69.6	260*	1844.65*	509.9*	4.32	74.8
12	4	PSO	608.45	2218.25	2.80	79.8*	250.75	2156.35	585	2.62*	72.7
13	5	DE	435.15	1983.25	4.45	70.2*	260	1952.15	441.55	4.54	64.45
14	5	DEFPSO	498.55	2249.4	2.54*	68.65	260	2193.25	483.45	2.76	66.45
15	5	MC	408.4*	1862.6*	4.06	63.35	260	1856.05*	420.65*	4.49	71.05*
16	5	PSO	492.35	2218.1	2.82	68.4	260	2203.65	488.9	2.6*	68.95
17	6	DE	369.85	2015.4	4.92	64.45	260	1980.15	379.7	4.25	61.1
18	6	DEFPSO	401.1	2144.1	3.25*	56.65	260	2149.85	406.25	3.18*	62.55
19	6	MC	352.95*	1909.7*	4.06	66.95*	260	1873.65*	363.9*	4.66	56.45
20	6	PSO	432.55	2283.65	3.33	60.3	260	2252.85	427.9	3.22	70.4*
21	7	DE	324.4	2015.9	4.43	53.25	260	1976.6	341.3	3.78	58.15
22	7	DEFPSO	345.65	2109.05	3.97	54.65	260	2160.35	357.8	3.38	50.5
23	7	MC	311.25*	1961.9*	4.36	53.8	260	1848.7*	320.4*	4.78	57.45
24	7	PSO	369.3	2246.25	3.39*	59.4*	260	2335.6	385.1	3.17*	61.4*
25	8	DE	278.95	1955.05*	5.97	46	260	1948.85	289.35	4.28	49.9
26	8	DEFPSO	313.7	2136.2	3.6	52.85*	260	2128	313.85	3.46	49.15
27	8	MC	277.9*	1980.6	4.75	48.7	260	1839.95*	280.95*	4.65	39.1
28	8	PSO	331.6	2266.35	3.4*	52.35	260	2279.4	332.95	2.95*	52.8*
29	9	DE	252.5	1991.25	4.98	43.35	260	1939.85	256.1*	5.09	41.65
30	9	DEFPSO	279.1	2135.55	3.21*	44.9	260	2153.05	283.35	3.09*	42.35
31	9	MC	251*	1965.9*	4.59	40.85	260	1891.35*	262.6	4.41	40.3
32	9	PSO	303.7	2264.45	4.37	44.95*	260	2165.7	290.05	3.97	50.75*
33	10	DE	229.75	2004.25	5.82	37.35	260	1917.55	233.7*	4.65	32.35
34	10	DEFPSO	262.75	2197.5	3.51*	41.7	260	2171.05	265.15	3.5*	37.7*
35	10	MC	228.85*	1985.4*	4.85	35.65	260	1904.45*	234.85	4.34	37.25
36	10	PSO	269.9	2235.25	4.26	41.9*	260	2176.95	266.55	3.8	37

DE: Differential Evolution; DEFPSO: Differential Evolution–Fused Particle Swarm Optimization; MC: Multi-strategy Coevolution; PSO: Particle Swarm Optimization.

The best values among the compared algorithms are marked with asterisks, and the ones yielded by MC are shaded (when the values from all algorithms are equal, no asterisk is given).

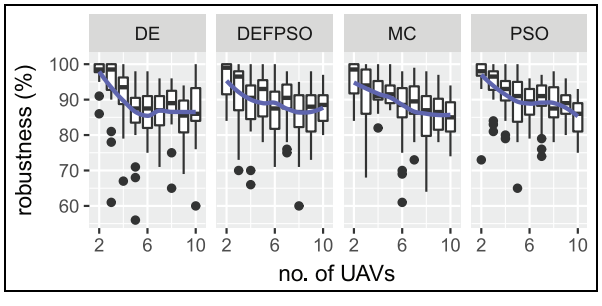

Figure 15. Robustness level of schedule based on simulations with $MS_{max} = 600$ and large number of competitive alternative paths.

given). Each mean value corresponds to the average of the values from the respective 20 simulation runs. Hence, in total there are $36 \times 20 = 720$ runs from each case (where $MS_{max} = 600$ s and $MS_{max} = \infty$).

Conclusion

This study investigates a delay-tolerant scheduling system for bridge inspection operations. A constructive algorithm called Retractable Chain Task Assignment algorithm is presented. RCTA is tailored for assigning slack time on the task schedules. RCTA is incorporated with a proposed metaheuristic algorithm called Multi-

strategy Coevolution to search the solution space. MC utilizes linked clone groups to perform multiple strategies separately and yet share the best solution throughout the entire search. Furthermore, the elite group (which is cloned to form the other groups) is replaced every several iterations through a tournament selection of the other groups. The generated schedules are compared against simulations to measure the robustness level. A diminishing benefit behavior as the number of UAVs increases is found, and the appropriate number of multiple UAVs in proportion to the scale of the tasks and environment (map) can vary. The effectiveness of the proposed method is proportional to the well-balanced scales of the map and the UAVs. Furthermore, MC outperforms the DE, PSO, and DEFPSO in terms of the schedule objectives (makespan and battery consumption). The robustness level of MC is on par with the other algorithms. In terms of computation time, MC is inferior toward PSO while being on par with the others. This is consistent on both cases: with and without the maximum makespan limitation.

Acknowledgements

The authors would like to thank Weikun Zhen for his prompt feedback during the flight data analysis.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has partly been supported by Innovation Fund Denmark under project UAWorld (grant agreement number 9-2014-3) and National Science Foundation–National Robotics Initiative (NSF-NRI) grant.

ORCID iD

Yohanes Khosiawan https://orcid.org/0000-0002-9729-8485

References

1. Coelho BN, Coelho VN, Coelho IM, et al. A multi-objective green UAV routing problem. *Comput Oper Res* 2017; 88: 306–315.
2. Wen T, Zhang Z and Wong KK. Multi-objective algorithm for blood supply via unmanned aerial vehicles to the wounded in an emergency situation. *PLoS ONE* 2016; 11: e0155176.
3. Fox MS and Sadeh NM. Why is scheduling difficult? A CSP perspective. In: *Proceedings of the ninth European conference on artificial intelligence (ECAI)*, Stockholm, 6 August 1990, pp.754–767. London: Pitman Publishing.
4. Khosiawan Y, Park Y, Moon I, et al. Task scheduling system for UAV operations in indoor environment. *Neural Comput Appl*. Epub ahead of print 23 February 2018. DOI: 10.1007/s00521-018-3373-9
5. Nigam N, Bieniawski S, Kroo I, et al. Control of multiple UAVs for persistent surveillance: algorithm and flight test results. *IEEE T Contr Syst T* 2012; 20: 1236–1251.
6. Verderame PM, Elia JA, Li J, et al. Planning and scheduling under uncertainty: a review across multiple sectors. *Ind Eng Chem Res* 2010; 49: 3993–4017.
7. Park Y, Khosiawan Y, Moon I, et al. Scheduling system for multiple unmanned aerial vehicles in indoor environments using the CSP approach. In: Czarnowski I, Caballero A, Howlett R, et al. (eds) *Intelligent decision technologies*. Cham, Springer, 2016, pp.77–87.
8. Nielsen I, Dang QV, Nielsen P, et al. Scheduling of mobile robots with preemptive tasks. In: *11th international conference on distributed computing and artificial intelligence*, Salamanca, 4–6 June 2014, pp.19–27. Cham: Springer.
9. Grewal MS, Weill LR and Andrews AP. *Global positioning systems, inertial navigation, and integration*. Hoboken, NJ: John Wiley & Sons, 2007.
10. Wu CW, Brown KN and Beck JC. Scheduling with uncertain release dates. In: *Irish conference on artificial intelligence and cognitive science (AICS '05)*, Portstewart, 7–9 September 2005, p.397. Coleraine: University of Ulster.
11. Li Z and Ierapetritou MG. Robust optimization for process scheduling under uncertainty. *Ind Eng Chem Res* 2008; 47: 4148–4157.
12. Herroelen W and Leus R. Project scheduling under uncertainty: survey and research potentials. *Eur J Oper Res* 2005; 165: 289–306.
13. Davenport A, Gefflot C and Beck C. Slack-based techniques for robust schedules. In: *Sixth European conference on planning*, Toledo, 12–14 September 2001, pp.43–49. Palo Alto, CA: AAAI Press.
14. Pongpunwattana A and Rysdyk R. Real-time planning for multiple autonomous vehicles in dynamic uncertain environments. *J Aeros Comp Inf Com* 2004; 1: 580–604.
15. Bertuccelli L, Choi HL, Cho P, et al. Real-time multi-UAV task assignment in dynamic and uncertain environments. In: *AIAA guidance, navigation, and control conference*, Chicago, IL, 10–13 August, pp.1–16. Reston, VA: AIAA.
16. Kaveh A. *Advances in metaheuristic algorithms for optimal design of structures*. Berlin: Springer, 2017.
17. Lin G, Zhao K and Wan Q. Takagi-Sugeno fuzzy model identification using coevolution particle swarm optimization with multi-strategy. *Appl Intell* 2016; 45: 187–197.
18. Shi J, Gong M, Ma W, et al. A multipopulation coevolutionary strategy for multiobjective immune algorithm. *Sci World J* 2014; 2014: 539128.

19. Valença J, Puenteú IJ, lio E, et al. Assessment of cracks on concrete bridges using image processing supported by laser scanning survey. *Constr Build Mater* 2017; 146: 668–678.
20. Graybeal B, Walther R and Washer G. Ultrasonic inspection of bridge hanger pins. *Transp Res Record* 2000; 1697: 19–23.
21. Garey MR and Johnson DS. *Computers and intractability*, Vol. 29. New York: W. H. Freeman, 2002.
22. Gen M, Cheng R and Lin L. *Network models and optimization: multiobjective genetic algorithm approach*. Berlin: Springer, 2008.
23. Quigley M, Conley K, Gerkey B, et al. ROS: an open-source robot operating system. In: *IEEE international conference on robotics and automation*, Kobe, Japan, 12–17 May 2009, p.5. New York: IEEE.
24. Khosiawan Y, Khalfay A and Nielsen I. Scheduling unmanned aerial vehicle and automated guided vehicle operations in an indoor manufacturing environment using differential evolution-fused particle swarm optimization. *Int J Adv Robot Syst* 2018; 15: 1–5.
25. Li Z, Janardhanan MN, Tang Q, et al. Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem. *Adv Mech Eng* 2016; 8: 1–14.
26. Cleveland WS, Grosse E and Shyu WM. Local regression models. In: Chambers JM and Hastie TJ (eds) *Statistical models in S*. Pacific Grove, CA: Wadsworth & Brooks/Cole, 1991, pp.309–376.