Journal of Systems Architecture 140 (2023) 102895

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Heuristic task scheduling on heterogeneous UAVs: A combinatorial optimization approach

Keqin Li

Department of Computer Science, State University of New York, New Paltz, NY, 12561, USA

ARTICLE INFO

Keywords:
Algorithmic framework
Combinatorial optimization
Distance-centric problems
Heterogeneous UAVs
Heuristic algorithm
Lower/upper bound
NP-hardness
Performance bound
Task scheduling
Time-centric problems

ABSTRACT

A fleet of unmanned aerial vehicles (UAVs) provide a new and unique type of distributed computing paradigm and platform. This is a distributed computing environment with mobile servers, where a server (i.e., a UAV) moves around to process tasks. Task scheduling for UAVs has several unique characteristics, such as heterogeneity, mobility, and locality. UAVs are heterogeneous in the sense that they have different initial positions, flight speeds, and execution times. While task assignment and flight planning have been studied extensively, there has been little research of task scheduling on heterogeneous UAVs within the framework of combinatorial optimization, i.e., heuristic algorithms for NP-hard problems and their performance evaluation when compared with optimal solutions. In this paper, we take a combinatorial optimization approach to addressing the issues in task scheduling for heterogeneous UAVs. The main contributions are summarized as follows. We define eight combinatorial optimization problems, i.e., four time-centric problems including the completion time minimization problem, the total time minimization problem, the finished tasks maximization with time constraint problem, the reward maximization with time constraint problem; and four distance-centric problems including the longest distance minimization problem, the total distance minimization problem, the finished tasks maximization with distance constraint problem, the reward maximization with distance constraint problem. We prove that all these problems are NP-hard. We develop two efficient and effective heuristic algorithmic frameworks to solve our problems, one for minimization problems and one for maximization problems. We derive lower/upper bounds for the optimal solutions. We evaluate the performance of our heuristic algorithms by comparing their solutions with optimal solutions and show that they are able to produce near-optimal solutions. To the best of the author's knowledge, this is the first paper which studies task scheduling on heterogeneous UAVs using a combinatorial optimization approach.

1. Introduction

1.1. Background and motivation

Unmanned aerial vehicles (UAVs) have been extensively used in military applications such as searching, recognizing, interfering, and attacking targets [1,2], reconnaissance missions [3], and civilian applications such as search and rescue missions [4,5], agricultural farming [6], forest fire detection [7], intelligent ocean control [8], and disaster relief [9].

A fleet of UAVs provide a new and unique type of distributed computing paradigm and platform. This is a distributed computing environment with mobile servers, where a server (i.e., a UAV) moves around to process tasks. Task scheduling for UAVs has several unique characteristics, such as heterogeneity, mobility, and locality. UAVs are heterogeneous in the sense that they have different initial positions, flight speeds, and execution times. A UAV is actually a mobile server, which moves to the position of a task to process the task, and then moves to the position of the next task. A UAV tends to serve tasks nearby rather than remote tasks to void long flight time and to reduce task processing time. (Notice that the above description includes multiple tasks at the same location and the same task distributed at various locations as special cases.)

There are two subproblems in task scheduling on UAVs [10–12]. The first subproblem is *task assignment*, i.e., to divide a set of tasks into disjoint subsets and to assign each subset of tasks to a UAV, so that the completion time of all tasks is minimized. This subproblem alone (when all UAVs and tasks are at the same location) is NP-hard even for two homogeneous UAVs (see Section 2.3). The second subproblem is *flight planning*, i.e., to decide a flight route (i.e., a sequence of tasks) for each UAV, so that the length of the route is minimized. This subproblem alone (when all task execution times are zero) is NP-hard even for one

E-mail address: lik@newpaltz.edu.

https://doi.org/10.1016/j.sysarc.2023.102895
Received 21 February 2023; Received in revised form 6 April 2023; Accepted 4 May 2023
Available online 10 May 2023

UAV to traverse all task locations, where each task is visited exactly once (see Section 2.3).

While task assignment and flight planning have been studied extensively, there has been little research of task scheduling on heterogeneous UAVs within the framework of combinatorial optimization, i.e., heuristic algorithms for NP-hard problems and their performance evaluation when compared with optimal solutions. The motivation of this paper is to explore in this direction.

As in all task scheduling systems, performance and cost are two main considerations. For task scheduling on UAVs, performance is mainly task processing time, and cost is essentially drone flight distance. We consider two types of combinatorial optimization problems. The first type is the class of *time-centric problems* (Section 4). In these problems, either processing time related metrics are optimized, or other metrics are optimized with processing time constraint. Time-centric optimization problems concentrate on scheduling performance enhancement. The second type is the class of *distance-centric problems* (Section 5). In these problems, either flight distance related metrics are optimized, or other metrics are optimized with flight distance constraint. Distance-centric optimization problems focus on resource consumption reduction. It is interesting and important to notice that task processing time and drone flight distance are positively correlated, since processing time includes flight time, which depends on the flight distance.

1.2. New contributions

In this paper, we take a combinatorial optimization approach to addressing the issues in task scheduling for heterogeneous UAVs. The main contributions are summarized as follows.

- We define eight combinatorial optimization problems, i.e., four time-centric problems including the completion time minimization (CTM) problem, the total time minimization (TTM) problem, the finished tasks maximization with time constraint (FTM-TC) problem, the reward maximization with time constraint (RM-TC) problem; and four distance-centric problems including the longest distance minimization (LDM) problem, the total distance minimization (TDM) problem, the finished tasks maximization with distance constraint (FTM-DC) problem, the reward maximization with distance constraint (RM-DC) problem. (Note: The reason we consider eight problems is that they can all be solved using our algorithmic frameworks.)
- We prove that all these problems are NP-hard.
- We develop two efficient and effective heuristic algorithmic frameworks to solve our problems, one for minimization problems (e.g., CTM, TTM, LDM, TDM) and one for maximization problems (e.g., FTM-TC, RM-TC, FTM-DC, RM-DC).
- We derive lower/upper bounds for the optimal solutions.
- We evaluate the performance of our heuristic algorithms by comparing their solutions with optimal solutions (actually, the lower/upper bounds) and show that they are able to produce near-optimal solutions.

To the best of the author's knowledge, this is the first paper which studies task scheduling on heterogeneous UAVs using a combinatorial optimization approach.

Our study in this paper has two unique features. First, we develop two algorithmic frameworks which are likely to be applicable to a variety of optimization problems of task scheduling on UAVs. Second, the performance of each heuristic algorithm is evaluated in such a way that its solution is compared with an optimal solution (actually, its lower or upper bound). Such an efficacious approach has not been seen in the existing literature for task scheduling on UAVs. Therefore, our investigation in this paper has made significant contributions to the field of task scheduling on heterogeneous UAVs.

The rest of the paper is organized as follows. In Section 2, we present preliminary discussion, including a task scheduling model on UAVs, a graph-theoretical formulation of a UAV, the traveling salesman path problem, and performance measures for heuristic algorithms. In Section 3, we develop our algorithmic frameworks, one for minimization problems and one for maximization problems. In Sections 4 and 5, we deal with time-centric optimization problems and distance-centric optimization problems respectively. For each problem, we give its definition, prove its NP-hardness, present a heuristic algorithm, derive a lower bound (for a minimization problem) or an upper bound (for a maximization problem), and evaluate its performance. In Section 6, we review related research. In Section 7, we conclude the paper.

2. Preliminaries

In this section, we present preliminary discussion, including a task scheduling model on UAVs, a graph-theoretical formulation of a UAV, the traveling salesman path problem, and performance measures for heuristic algorithms.

2.1. Task scheduling on UAVs

In this section, we describe our task scheduling model on heterogeneous UAVs (see Fig. 1 for an illustration).

Let $U = \{u_1, u_2, \dots, u_m\}$ be a set of heterogeneous UAVs, and $T = \{t_1, t_2, \dots, t_n\}$ be a set of tasks. For all $1 \le i \le m$ and $1 \le j, j' \le n$, we define the following quantities. (The Appendix provides a list of notations and their definitions in the order introduced in this paper.)

- $pos(u_i) = (x(u_i), y(u_i), z(u_i))$ is the position (i.e., the initial location) of u_i.
- $pos(t_j) = (x(t_j), y(t_j), z(t_j))$ is the position of t_j.
- $dist(u_i, t_j)$ is the distance between $pos(u_i)$ and $pos(t_j)$.
- $dist(t_j, t_{j'})$ is the distance between $pos(t_j)$ and $pos(t_{j'})$.
- $vel(u_i)$ is the flight velocity of u_i.
- $ftime(u_i, t_j) = dist(u_i, t_j)/vel(u_i)$ is the flight time of u_i from $pos(u_i)$ to $pos(t_j)$.
- $ftime_i(t_j, t_{j'}) = dist(t_j, t_{j'})/vel(u_i)$ is the flight time of u_i from $pos(t_j)$ to $pos(t_{j'})$.
- $etime(u_i, t_j)$ is the execution time of u_i to process t_j.
- $reward(t_j)$ is the reward (related to urgency, priority, deadline, etc.) to finish t_j.

All positions are fixed and do not change. However, UAVs change their locations (see Section 3).

The UAVs are heterogeneous in the sense that they have different flight velocities and execution speeds. The UAVs are homogeneous if they have the same flight velocity and identical execution time for the same task, i.e., $vel(u_i) = vel$ and $etime(u_i, t_j) = etime(t_j)$, for all $1 \le i \le m$ and $1 \le j \le n$.

In general, due to the sophistication of a real physical environment (e.g., objects and obstacles in between), we do not make any assumption on the distance between two identities. The distance between two identities can be asymmetric, e.g., $dist(t_j, t_{j'}) \ne dist(t_{j'}, t_j)$.

The distance between two positions may not be analytically related to their positions. As a special case, the distance between two positions $pos = (x, y, z)$ and $pos' = (x', y', z')$ can be Euclidean, i.e.,

$$dist(pos, pos') = \sqrt{(x - x')^2 + (y - y')^2 + (z - z')^2}.$$

Let $T_i = \{t_{j_1}, t_{j_2}, \dots, t_{j_{n_i}}\}$ be a set of tasks assigned to u_i. Note that $(T_1, T_2, \dots, T_m)$, where the T_i's are disjoint and $T_1 \cup T_2 \cup \cdots \cup T_m = T$, is actually a *schedule* of T on the m UAVs. For convenience, $T_i = (t_{j_1}, t_{j_2}, \dots, t_{j_{n_i}})$ is also treated as a sequence of tasks, so that we can get a flight route of u_i: $(u_i, t_{j_1}, t_{j_2}, \dots, t_{j_{n_i}})$. Hence, a schedule is a solution to the subproblems of task assignment and flight planning simultaneously.

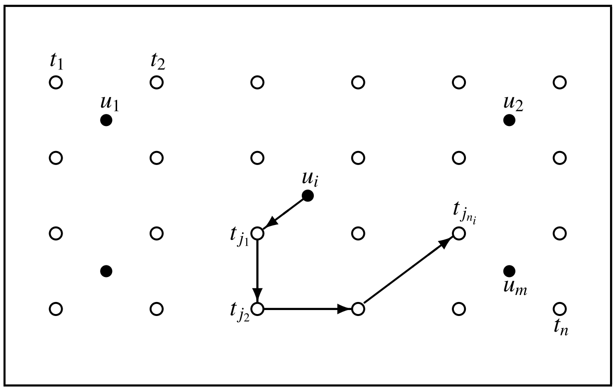

Fig. 1. Task scheduling on heterogeneous UAVs.

The processing time of a task is the flight time to reach the task plus its execution time, i.e.,

$$ptime(t_{j_1}) = ftime(u_i, t_{j_1}) + etime(u_i, t_{j_1}),$$

and

$$ptime(t_{j_k}) = ftime_i(t_{j_{k-1}}, t_{j_k}) + etime(u_i, t_{j_k}),$$

for all $2 \le k \le n_i$.

The time for u_i to process tasks in T_i is the total processing time, i.e.,

$$time(u_i) = \sum_{k=1}^{n_i} ptime(t_{j_k}),$$

which is the total flight time plus the total execution time, i.e.,

$$time(u_i) = ftime(u_i, t_{j_1}) + \sum_{k=2}^{n_i} ftime_i(t_{j_{k-1}}, t_{j_k}) + \sum_{k=1}^{n_i} etime(u_i, t_{j_k}),$$

which is

$$time(u_i) = \frac{1}{vel(u_i)}\left(dist(u_i, t_{j_1}) + \sum_{k=2}^{n_i} dist(t_{j_{k-1}}, t_{j_k})\right) + \sum_{k=1}^{n_i} etime(u_i, t_{j_k}),$$

and

$$time(u_i) = \frac{distance(u_i)}{vel(u_i)} + \sum_{k=1}^{n_i} etime(u_i, t_{j_k}),$$

where

$$distance(u_i) = dist(u_i, t_{j_1}) + \sum_{k=2}^{n_i} dist(t_{j_{k-1}}, t_{j_k})$$

is the flight distance of u_i.

The completion time of T is

$$ctime(T) = \max\{time(u_1), time(u_2), \dots, time(u_m)\},$$

which is the maximum processing time of all UAVs, and $ctime(T)$ is actually the *makespan* of the schedule. The total time of T is

$$ttime(T) = time(u_1) + time(u_2) + \cdots + time(u_m),$$

which is the total processing time of all UAVs.

The longest distance of T is

$$ldistance(T) = \max\{distance(u_1), distance(u_2), \dots, distance(u_m)\},$$

which is the longest flight distance of all UAVs. The total distance of T is

$$tdistance(T) = distance(u_1) + distance(u_2) + \cdots + distance(u_m),$$

which is the total flight distance of all UAVs.

The reward of u_i is the total reward of tasks in T_i:

$$reward(u_i) = \sum_{k=1}^{n_i} reward(t_{j_k}).$$

2.2. A graph-theoretical formulation

In this section, we give a graph-theoretical formulation of a UAV, which is applicable to all our combinatorial optimization problems.

Let us consider a weighted directed graph $G = (U \cup T, E, dist)$, where $U \cup T$ is the set of vertices, $E = (U \cup T) \times T$ is the set of edges, and $dist$ gives the weights of the edges. UAV flight velocities vel and task execution times $etime$ are ignored here, or, one can simply assume that $vel(u_i) = 1$ and $etime(u_i, t_j) = 0$, for all $1 \le i \le m$ and $1 \le j \le n$. For a set (or sequence) of vertices $T_i = (t_{j_1}, t_{j_2}, \dots, t_{j_{n_i}})$ assigned to u_i, the length of the path (i.e., flight route) $P_i = (u_i, t_{j_1}, t_{j_2}, \dots, t_{j_{n_i}})$ is

$$length(P_i) = dist(u_i, t_{j_1}) + \sum_{k=2}^{n_i} dist(t_{j_{k-1}}, t_{j_k}),$$

which is actually the same as $time(u_i)$ and $distance(u_i)$.

Our combinatorial optimization problems are typically to determine a partition of T into m disjoint subsets $T_1, T_2, \dots, T_m$, where T_i is assigned to u_i and $T_1 \cup T_2 \cup \cdots \cup T_m = T$, such that the longest path length, i.e.,

$$llength(T) = \max_{1\le i\le m}\{length(P_i)\},$$

or the total path length, i.e.,

$$tlength(T) = \sum_{i=1}^{m} length(P_i),$$

is minimized.

When $m = 1$, i.e., there is only one UAV and $U = \{u\}$, minimizing both $llength(T)$ and $tlength(T)$ is equivalent to determining a permutation of T, i.e., $(t_{j_1}, t_{j_2}, \dots, t_{j_n})$, such that the length of the path $P = (u, t_{j_1}, t_{j_2}, \dots, t_{j_n})$ is minimized.

Given a weighted directed graph $G = (V, E, dist)$, where $V = \{v_0, v_1, v_2, \dots, v_n\}$, a *traveling salesman path* starting from v_0 is $P = (v_0, v_{j_1}, v_{j_2}, \dots, v_{j_n})$, which is actually a permutation of $v_1, v_2, \dots, v_n$ (i.e., $(v_{j_1}, v_{j_2}, \dots, v_{j_n})$), such that each vertex in the graph is visited exactly once. We will show that it is NP-hard to find a shortest traveling salesman path, which forms the basis of the NP-hardness of all our combinatorial optimization problems.

2.3. The traveling salesman path problem

In this section, we prove the NP-hardness of the traveling salesman path problem.

For a weighted directed graph $G = (V, E, dist)$, where $V = \{v_1, v_2, \dots, v_n\}$, a *traveling salesman loop* is $L = (v_{j_1}, v_{j_2}, \dots, v_{j_n}, v_{j_1})$, which is actually a permutation of $v_1, v_2, \dots, v_n$ (i.e., $(v_{j_1}, v_{j_2}, \dots, v_{j_n})$), such that each vertex in the graph is visited exactly once. Given a weighted directed graph $G = (V, E, dist)$, the *traveling salesman problem* (TSP) is to find a traveling salesman loop $L = (v_{j_1}, v_{j_2}, \dots, v_{j_n}, v_{j_1})$, such that the length of the loop, i.e.,

$$length(L) = \sum_{k=1}^{n-1} dist(v_{j_k}, v_{j_{k+1}}) + dist(v_{j_n}, v_{j_1}),$$

is minimized.

Given two vertices v_j and $v_{j'}$, a traveling salesman path between the two vertices is a path $P = (v_j, v_{j_1}, \dots, v_{j_{n-2}}, v_{j'})$, where $(j_1, \dots, j_{n-2})$ is a permutation of the indices in $\{1, 2, \dots, n\} - \{j, j'\}$. Given a weighted directed graph $G = (V, E, dist)$ and two vertices v_j and $v_{j'}$, the *traveling salesman path problem* (TSPP) is to find a traveling salesman path $P = (v_j, v_{j_1}, \dots, v_{j_{n-2}}, v_{j'})$, such that the length of the path, i.e.,

$$length(P) = dist(v_j, v_{j_1}) + \sum_{k=1}^{n-3} dist(v_{j_k}, v_{j_{k+1}}) + dist(v_{j_{n-2}}, v_{j'}),$$

is minimized.

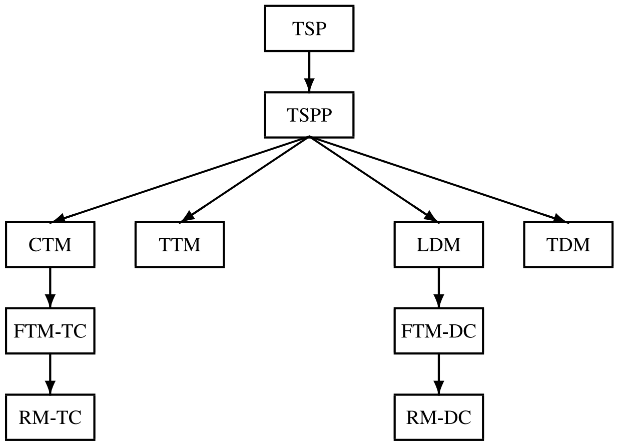

Fig. 2. Reductions among the problems in their NP-hardness proofs.

Theorem 0. *The TSPP problem is NP-hard.*

Proof. We show that the TSPP is NP-hard by a Turing reduction (Cook reduction) from the TSP, which is well known to be NP-hard ([13], p. 211). Consider any edge $(v_{j'}, v_j)$. If $(v_{j'}, v_j)$ is on a traveling salesman loop, then we only need to find a traveling salesman path between v_j and $v_{j'}$, i.e., $P = (v_j, v_{j_1}, \ldots, v_{j_{n-2}}, v_{j'})$, with the minimum length. The path can be extended to a traveling salesman loop $L = (v_j, v_{j_1}, \ldots, v_{j_{n-2}}, v_{j'}, v_j)$ with

$$length(L) = length(P) + dist(v_{j'}, v_j).$$

Hence, if there is a polynomial time algorithm A for the TSPP, then we can run the algorithm for each edge $(v_{j'}, v_j)$, obtain a traveling salesman path $P = (v_j, v_{j_1}, \ldots, v_{j_{n-2}}, v_{j'})$, and calculate the length of the loop $L = (v_j, v_{j_1}, \ldots, v_{j_{n-2}}, v_{j'}, v_j)$. After executing algorithm A for $n(n-1)$ times, once for each edge, we can choose the loop with the minimum length, and solve the TSP. If A has polynomial time complexity $T(n)$, then the above algorithm for the TSP has time complexity $O(n^2 T(n))$, which is also a polynomial. □

Fig. 2 shows reductions among the problems in their NP-hardness proofs. We would like to emphasize that all our combinatorial optimization problems studied in this paper remain NP-hard for a single UAV, homogeneous UAVs, and zero execution times. Furthermore, since the TSP is also NP-hard for weighted undirected graphs ([13], p. 211) and Euclidean distances ([13], p. 212), so are our problems. We also mention that even when all UAVs and tasks are at the same position (i.e., no movement is required), all our combinatorial optimization problems remain NP-hard for two homogeneous UAVs. This can be proved using a reduction from the *multiprocessor scheduling* problem ([13], p. 238).

2.4. Performance measures

In this section, we define our performance measures for heuristic algorithms.

Let $A(I)$ be the solution of a heuristic algorithm A for an instance I, and Opt(I) be the optimal solution of I.

For a minimization problem, we say that B is a *performance bound* if

$$\frac{A(I)}{\mathrm{Opt}(I)} \le B,$$

for all I. When I is randomized, if

$$\boldsymbol{E}\left[\frac{A(I)}{\mathrm{Opt}(I)}\right] \le B,$$

B is an *expected performance bound.*

For a maximization problem, we say that B is a *performance bound* if

$$\frac{A(I)}{\mathrm{Opt}(I)} \ge B,$$

for all I. When I is randomized, if

$$\boldsymbol{E}\left[\frac{A(I)}{\mathrm{Opt}(I)}\right] \ge B,$$

B is an *expected performance bound.*

3. Algorithmic frameworks

In this section, we develop our algorithmic frameworks, one for minimization problems (e.g., CTM, TTM, LDM, TDM) and one for maximization problems (e.g., FTM-TC, RM-TC, FTM-DC, RM-DC).

(Note: We present our algorithmic frameworks before we define our combinatorial optimization problems, since all our heuristic algorithms follow these frameworks. The reader may read Sections 4.s.1 and 5.s.1 ($1 \le s \le 4$) for problem definitions and then read this section. Section 3.1 should be read together with Sections 4.1.3, 4.2.3, 5.1.3, 5.2.3, and 3.2 should be read together with Sections 4.3.3, 4.4.3, 5.3.3, 5.4.3).

3.1. An algorithmic framework for minimization problems

In this section, we develop an algorithmic framework for all our minimization problems.

Algorithmic Framework for Minimization Problems (AF-Min)

```
for ($i = 1$; $i \le m$; $i$++) do                                        (1)
    $T_i \leftarrow \emptyset$;                                                      (2)
    $time(u_i) \leftarrow 0$;                                                (3)
    $distance(u_i) \leftarrow 0$;                                            (4)
    $loc(u_i) \leftarrow 0$;                                                 (5)
end do;                                                                   (6)
for ($j = 1$; $j \le n$; $j$++) do                                        (7)
    $best(t_j) \leftarrow \mathrm{argmin}_{1 \le i \le m}$\{problem-dependent expression\};   (8)
end do;                                                                   (9)
while ($T \ne \emptyset$) do                                              (10)
    $t_j \leftarrow \mathrm{argmin}_{t_{j'} \in T}$\{problem-dependent expression\};          (11)
    $i \leftarrow best(t_j)$;                                                (12)
    $T_i \leftarrow T_i \cup \{j\}$;                                         (13)
    $T \leftarrow T - \{j\}$;                                                (14)
    $time(u_i) \leftarrow time(u_i) + ptime(u_i, t_j)$;                      (15)
    $distance(u_i) \leftarrow distance(u_i) + fdist(u_i, t_j)$;              (16)
    $loc(u_i) \leftarrow j$;                                                 (17)
    for (each $t_{j'} \in T$) do                                          (18)
        $best(t_{j'}) \leftarrow \mathrm{argmin}_{1 \le i \le m}$\{problem-dependent exp\};   (19)
    end do;                                                               (20)
end do;                                                                   (21)
return $T_1, T_2, \ldots, T_m$.                                           (22)
```

Each u_i is a mobile server. The current location of u_i is $loc(u_i) \in \{0, 1, 2, \ldots, n\}$. Initially, $loc(u_i) = 0$ (line (5)), which means that u_i is at $pos(u_i)$. After u_i reaches t_j and processes t_j, $loc(u_i) = j$ (line (17)), which means that u_i is at $pos(t_j)$.

Let $ptime(u_i, t_j)$ denote the processing time of u_i for t_j, which is

$$ptime(u_i, t_j) = \begin{cases} ftime(u_i, t_j) + etime(u_i, t_j), & \text{if } loc(u_i) = 0; \\ ftime_i(t_{loc(u_i)}, t_j) + etime(u_i, t_j), & \text{if } loc(u_i) \ne 0. \end{cases}$$

Let $fdist(u_i, t_j)$ denote the flight distance from u_i to t_j, which is

$$fdist(u_i, t_j) = \begin{cases} dist(u_i, t_j), & \text{if } loc(u_i) = 0; \\ dist(t_{loc(u_i)}, t_j), & \text{if } loc(u_i) \ne 0. \end{cases}$$

Let T_i be the set of tasks allocated to u_i so far. Let $time(u_i)$ be the total processing time of tasks in T_i and $distance(u_i)$ be the total flight distance for tasks in T_i. Initially, $T_i = \emptyset$, $time(u_i) = 0$, and $distance(u_i) = 0$ (lines (2)–(4)).

Each task t_j has a $best(t_j)$ value, where $best(t_j) \in \{1, 2, \dots, m\}$, which gives the best u_i that processes t_j, and is initialized in line (8) and updated in line (19). The definition of $best(t_j)$ is problem-dependent.

The greedy algorithm repeatedly chooses the next task to process (lines (10)–(21)). The next task t_j is chosen in such a way that it has the minimum $best(t_j)$ value (line (11)). Task t_j is assigned to the corresponding u_i (lines (12)–(14)). Then, the total processing time, the total flight distance, and the location of u_i are updated (lines (15)–(17)). Once u_i moves to t_j, the $best(t_{j'})$ value of each remaining task $t_{j'}$ is possibly modified (lines (18)–(20)).

Note: All problem-dependent expressions in lines (8), (11), (19) will be specified in Sections 4.1.3, 4.2.3, 5.1.3, 5.2.3 for specific minimization problems.

The time complexity of AF-Min is analyzed as follows. The for-loop in lines (1)–(6) takes $O(m)$ time. The for-loop in lines (7)–(9) takes $O(mn)$ time. The for-loop in lines (10)–(21) is repeated n times. Line (11) takes $O(n)$ time. Line (19) takes $O(m)$ time, and the for-loop in lines (18)–(20) takes $O(mn)$ time. Therefore, the for-loop in lines (10)–(21) takes $O(mn^2)$ time. The overall time complexity of AF-Min is $O(mn^2)$.

3.2. An algorithmic framework for maximization problems

Algorithmic Framework for Maximization Problems (AF-Max)

```
for ($i = 1$; $i \le m$; $i$++) do                                          (1)
    $T_i \leftarrow \emptyset$;                                             (2)
    $time(u_i) \leftarrow 0$;                                               (3)
    $distance(u_i) \leftarrow 0$;                                           (4)
    $loc(u_i) \leftarrow 0$;                                                (5)
end do;                                                                     (6)
for ($j = 1$; $j \le n$; $j$++) do                                          (7)
    $uav(t_j) \leftarrow \{i \mid$ problem-dependent condition$\}$;         (8)
    $best(t_j) \leftarrow$ argmin/argmax$_{i \in uav(t_j)}$
                {problem-dependent expression}, if $uav(t_j) \ne \emptyset$;   (9)
end do;                                                                     (10)
while ($T \ne \emptyset$) do                                                (11)
    if ($uav(t_j) = \emptyset$ for all $t_j \in T$) then break;             (12)
    $t_j \leftarrow$ argmin/argmax$_{t_{j'} \in T \text{ and } uav(t_{j'}) \ne \emptyset}$
                      {problem-dependent expression};                       (13)
    $i \leftarrow best(t_j)$;                                               (14)
    $T_i \leftarrow T_i \cup \{j\}$;                                        (15)
    $T \leftarrow T - \{j\}$;                                               (16)
    $time(u_i) \leftarrow time(u_i) + ptime(u_i, t_j)$;                     (17)
    $distance(u_i) \leftarrow distance(u_i) + fdist(u_i, t_j)$;             (18)
    $loc(u_i) \leftarrow j$;                                                (19)
    for (each $t_{j'} \in T$) do                                            (20)
        if (problem-dependent condition) then                               (21)
            $uav(t_{j'}) \leftarrow uav(t_{j'}) \cup \{i\}$;                (22)
        else                                                                (23)
            $uav(t_{j'}) \leftarrow uav(t_{j'}) - \{i\}$;                   (24)
        $best(t_{j'}) \leftarrow$ argmin/argmax$_{i \in uav(t_{j'})}$
                    {problem-dependent exp}, if $uav(t_{j'}) \ne \emptyset$;   (25)
    end do;                                                                 (26)
end do;                                                                     (27)
return $T_1, T_2, ..., T_m$.                                                (28)
```

In this section, we develop an algorithmic framework for all our maximization problems.

Due to various performance and resource constraints in our maximization problems, not every u_i can process every t_j. Therefore, each task t_j has a $uav(t_j)$ value, which is the set of u_i's that can still accommodate t_j within its performance or resource constraint. The $uav(t_j)$ value is initialized in line (8) and updated in lines (21)–(24). The definition of $uav(t_j)$ is problem-dependent.

AF-Max follows the same structure as that of AF-Min. In each repetition of the loop in lines (11)–(27), we first check whether there is still any task which can be accommodated within the performance or resource constraint (line (12)). If so, the task t_j which has the minimum or maximum $best(t_j)$ value (line (13)) is assigned to the corresponding u_i (lines (14)–(19)). Then, the $uav(t_{j'})$ and $best(t_{j'})$ values of each remaining task $t_{j'}$ are possibly modified (lines (20)–(26)). The algorithm terminates if no task can be accommodated any more (line (12)) or all tasks are finished (line (11)).

Note: All problem-dependent conditions in lines (8), (21), and all problem-dependent expressions in lines (9), (13), (25) will be specified in Sections 4.3.3, 4.4.3, 5.3.3, 5.4.3 for specific maximization problems.

The time complexity of AF-Max is analyzed as follows. The for-loop in lines (1)–(6) takes $O(m)$ time. The for-loop in lines (7)–(10) takes $O(mn)$ time. The for-loop in lines (11)–(27) is repeated at most n times. Lines (12)–(13) take $O(n)$ time. Line (25) takes $O(m)$ time, and the for-loop in lines (20)–(26) takes $O(mn)$ time. Therefore, the for-loop in lines (11)–(27) takes $O(mn^2)$ time. The overall time complexity of AF-Max is $O(mn^2)$.

4. Time-centric problems

In this section, we deal with time-centric optimization problems, including the completion time minimization problem, the total time minimization problem, the finished tasks maximization with time constraint problem, and the reward maximization with time constraint problem. For each problem, we give its definition, prove its NP-hardness, present a heuristic algorithm, derive a lower bound (for a minimization problem) or an upper bound (for a maximization problem), and evaluate its performance.

4.1. Completion Time Minimization (CTM)

In this section, we address the completion time minimization problem.

4.1.1. Problem definition

The completion time minimization problem is to minimize the completion time of a set of tasks.

Problem 1 (*Completion Time Minimization*)**.** Given a set of heterogeneous UAVs $U = \{u_1, u_2, \dots, u_m\}$, and a set of tasks $T = \{t_1, t_2, \dots, t_n\}$, the *completion time minimization* (CTM) problem is to determine a partition of T into m disjoint subsets $T_1, T_2, \dots, T_m$, where T_i is assigned to u_i and $T_1 \cup T_2 \cup \cdots \cup T_m = T$, such that the completion time of T, i.e.,

$$ctime(T) = \max_{1 \le i \le m} \{time(u_i)\},$$

is minimized.

4.1.2. NP-hardness

Theorem 1A. *The CTM problem is NP-hard.*

Proof. We show the NP-hardness of the CTM problem using a Karp reduction (polynomial-time reduction) from the TSPP. Recall from the discussion in Section 2.2 that given a weighted directed graph $G = (V, E, dist)$ and a starting vertex v_j, our special case of the CTM problem (i.e., $m = 1$, $vel(u_i) = 1$ and $etime(u_i, t_j) = 0$, for all $1 \le i \le m$ and $1 \le j \le n$) is to find a traveling salesman path starting from v_j, i.e., $P = (v_j, v_{j_1}, \dots, v_{j_{n-1}})$ with the minimum length. This problem is similar but not identical to the TSPP, since TSPP specifies v_j and $v_{j'}$ (i.e., a path must start from v_j and finish at $v_{j'}$), and CTM only specifies v_j (i.e., a path must start from v_j but finish at any vertex).

Given an instance of the TSPP, i.e., $G = (V, E, dist)$ and v_j and $v_{j'}$, we construct an instance of the CTM problem, i.e., $G' = (V \cup v, E \cup \{v_{j'}, v\}, dist')$ and v_j, where $dist'$ is the same as $dist$ with the following extension, i.e., $dist'(v_{j'}, v) = 0$. Then, a traveling salesman path in G' starting at v_j with the minimum length must be $P' = (v_j, \ldots, v_{j'}, v)$. The reason is that if $(v_{j'}, v)$ is not the last edge in a path, the path looks like $(v_j, \ldots, v_{j_k}, v, v_{j_{k+1}}, \ldots, v_{j'})$, and there must be two edges (v_{j_k}, v) and $(v, v_{j_{k+1}})$; however, these edges do not even exist in G'. Furthermore, the path $P = (v_j, \ldots, v_{j'})$ must be the shortest path from v_j to $v_{j'}$ in G', which is also the shortest path from v_j to $v_{j'}$ in G. In addition, we have $length(P) = length(P')$. Hence, finding a shortest traveling salesman path from v_j to $v_{j'}$ in G is equivalent to finding a shortest traveling salesman path starting from v_j in G'. If the CTM problem can be solved in polynomial time $T(n)$, so is TSPP.

To summarize, we have shown a Karp reduction from the TSPP (and a Turing reduction from the TSP) to the CTM problem. Any polynomial time algorithm with time complexity $T(n)$ for the CTM problem can be used to solve the TSPP with the same time complexity, and can also be used to solve the TSP with time complexity $O(n^2 T(n))$. This completes the proof of the theorem. □

4.1.3. A heuristic algorithm

Our heuristic algorithm to solve the CTM problem is presented in Algorithm 1.

We define

$$best(t_j) = \operatorname*{argmin}_{1 \le i \le m} \{time(u_i) + ptime(u_i, t_j)\},$$

which decides the u_i, such that if u_i processes t_j, the new processing time of u_i is the shortest among all UAVs (line (19)). Initially (line (8)),

$$best(t_j) = \operatorname*{argmin}_{1 \le i \le m} \{ftime(u_i, t_j) + etime(u_i, t_j)\}.$$

The key idea of Algorithm 1 is in line (11), i.e., the next task t_j is chosen and assigned to $u_{best(t_j)}$, such that the new processing time of $u_{best(t_j)}$ (after t_j is processed) is the minimum among all remaining tasks. This can make $ctime(T)$ to increase slowly.

Algorithm 1: Completion Time Minimization

```
Input: $U = \{u_1, u_2, ..., u_m\}$, $T = \{t_1, t_2, ..., t_n\}$, and $dist(u_i, t_j)$, $dist(t_j, t_{j'})$,
$vel(u_i)$, $etime(u_i, t_j)$, for all $1 \le i \le m$ and $1 \le j, j' \le n$.
Output: $T_1, T_2, ..., T_m$, such that $ctime(T)$ is minimized.

The algorithm follows AF-Min with the following details in lines (8),
(11), (19):
    $best(t_j) \leftarrow \text{argmin}_{1 \le i \le m}\{ftime(u_i, t_j) + etime(u_i, t_j)\}$;              (8)
    $t_j \leftarrow \text{argmin}_{t_{j'} \in T}\{time(u_{best(t_{j'})}) + ptime(u_{best(t_{j'})}, t_{j'})\}$;   (11)
      $best(t_{j'}) \leftarrow \text{argmin}_{1 \le i \le m}\{time(u_i) + ptime(u_i, t_{j'})\}$;              (19)
```

4.1.4. A lower bound

Let $ptime^*(t_j)$ be the minimum possible processing time of t_j. For heterogeneous UAVs,

$$ptime^*(t_j) = \min_{1 \le i \le m} \left\{ \frac{1}{vel(u_i)} \min\left\{ \min_{1 \le i \le m} \{dist(u_i, t_j)\}, \min_{j' \ne j}\{dist(t_{j'}, t_j)\} \right\} + etime(u_i, t_j) \right\}.$$

For homogeneous UAVs,

$$ptime^*(t_j) = \frac{1}{vel} \min\left\{ \min_{1 \le i \le m} \{dist(u_i, t_j)\}, \min_{j' \ne j}\{dist(t_{j'}, t_j)\} \right\} + etime(t_j).$$

Table 1A
Simulation results of completion time minimization (Homogeneous UAVs, 99% confidence interval = ±0.64542%).

n	$\tau = 30$	$\tau = 50$	$\tau = 70$	$\tau = 90$
10	1.36479	1.27578	1.23612	1.21343
20	1.26453	1.20056	1.16705	1.14718
30	1.21928	1.16558	1.13613	1.12090
40	1.19118	1.14534	1.12168	1.10443
50	1.17086	1.13277	1.10961	1.09579
60	1.15842	1.12286	1.10142	1.08769
70	1.14756	1.11417	1.09571	1.08233
80	1.14059	1.10901	1.08964	1.07836
90	1.13309	1.10317	1.08604	1.07416
100	1.12753	1.09808	1.08269	1.07178

Table 1B
Simulation results of completion time minimization (Heterogeneous UAVs, 99% confidence interval = ±0.72353%).

n	$\tau = 30$	$\tau = 50$	$\tau = 70$	$\tau = 90$
10	1.57959	1.47121	1.41077	1.38262
20	1.43992	1.35168	1.30632	1.27778
30	1.37605	1.29843	1.25561	1.22829
40	1.33650	1.26203	1.22473	1.20242
50	1.30944	1.23744	1.20254	1.18047
60	1.28857	1.22151	1.18741	1.16466
70	1.27148	1.20617	1.17343	1.15324
80	1.25785	1.19658	1.16398	1.14467
90	1.24708	1.18704	1.15646	1.13642
100	1.23656	1.17766	1.14943	1.13068

Theorem 1B. *A lower bound for the optimal solution of the CTM problem is*

$$Opt(I) \ge ctime^*(T) = \frac{1}{m} \sum_{j=1}^{n} ptime^*(t_j).$$

Proof. The above lower bound can be justified as follows. The optimal completion time cannot be shorter than the total processing time divided by m, while the total processing time cannot be shorter than

$$\sum_{j=1}^{n} ptime^*(t_j),$$

since the processing time of t_j is at least $ptime^*(t_j)$. □

Let I be an instance of the CTM problem. Then, we have

$$\frac{A_1(I)}{\text{Opt}(I)} \le \frac{A_1(I)}{ctime^*(T)}.$$

Therefore,

$$B_1 = \boldsymbol{E}\left[\frac{A_1(I)}{ctime^*(T)}\right]$$

can be considered as an expected performance bound of Algorithm 1.

4.1.5. Performance evaluation

We consider a UAV task processing environment with $m = 5$ UAVs. All UAVs are randomly, uniformly, and independently distributed in a 1000 m × 1000 m × 200 m three dimensional space. The UAV flight velocities are randomly, uniformly, and independently distributed in the range $[20, 30]$ m/s, equivalent to approximately $[45, 67]$ mph.

All tasks are also randomly, uniformly, and independently distributed in a 1000 m × 1000 m × 200 m three dimensional space. Task execution times are randomly, uniformly, and independently distributed in the range $[\tau, 2\tau]$, where $\tau = 30, 50, 70, 90$ s. Task rewards are randomly, uniformly, and independently distributed in $\{1, 2, 3, \ldots, 10\}$.

We use the Euclidean distance for any two positions.

Tables 1A and 1B demonstrate our simulation results of completion time minimization for both homogeneous and heterogeneous UAVs respectively. The number of tasks is $n = 10, 20, \ldots, 100$. For each combination of n and τ, we generate $M = 1000$ random instances of the CTM problem. For each instance I, we apply Algorithm 1 to obtain $A_1(I)$, calculate the lower bound $ctime^*(T)$, and record the ratio $A_1(I)/ctime^*(T)$. The average of the M ratios is reported as B_1, with 99% confidence interval of ±0.64542% and ±0.72353% respectively. We have the following important observations.

- The performance bound B_1 is very close to one. This means that Algorithm 1 is able to efficiently find an assignment of the n tasks to the m UAVs and a flight route for each UAV, and to effectively reduce the complete time.
- The performance bound B_1 is closer to one as the number of tasks increases and/or as the task execution times become longer. This means that when there are many tasks, Algorithm 1 can more effectively and efficiently minimize the complete time. Furthermore, as task execution times become longer, since the impact of route planning becomes smaller, Algorithm 1 can perform better.
- Algorithm 1 performs better for homogeneous UAVs than heterogeneous UAVs, primarily due to less randomness and variability of flight velocities and execution times.

4.2. Total Time Minimization (TTM)

In this section, we address the total time minimization problem.

4.2.1. Problem definition

The total time minimization problem is to minimize the total time of a set of tasks.

Problem 2 (*Total Time Minimization*). Given a set of heterogeneous UAVs $U = \{u_1, u_2, \dots, u_m\}$, and a set of tasks $T = \{t_1, t_2, \dots, t_n\}$, the *total time minimization* (TTM) problem is to determine a partition of T into m disjoint subsets $T_1, T_2, \dots, T_m$, where T_i is assigned to u_i and $T_1 \cup T_2 \cup \dots \cup T_m = T$, such that the total time of T, i.e.,

$$ttime(T) = \sum_{i=1}^{m} time(u_i),$$

is minimized.

4.2.2. NP-hardness

Theorem 2A. *The TTM problem is NP-hard.*

Proof. When $m = 1$, the TTM problem is identical to the CTM problem, which is already known to be NP-hard when $m = 1$ using a reduction from the TSPP. □

4.2.3. A heuristic algorithm

Our heuristic algorithm to solve the TTM problem is presented in Algorithm 2.

```
Algorithm 2: Total Time Minimization

Input: $U = \{u_1, u_2, ..., u_m\}$, $T = \{t_1, t_2, ..., t_n\}$, and $dist(u_i, t_j)$, $dist(t_j, t_{j'})$, $vel(u_i)$, $etime(u_i, t_j)$, for all $1 \le i \le m$ and $1 \le j, j' \le n$.
Output: $T_1, T_2, ..., T_m$, such that $ttime(T)$ is minimized.

The algorithm follows AF-Min with the following details in lines (8), (11), (19):
    $best(t_j) \leftarrow \mathrm{argmin}_{1\le i\le m}\{ftime(u_i, t_j) + etime(u_i, t_j)\}$;    (8)
    $t_j \leftarrow \mathrm{argmin}_{t_{j'}\in T}\{ptime(u_{best(t_{j'})}, t_{j'})\}$;    (11)
        $best(t_{j'}) \leftarrow \mathrm{argmin}_{1\le i\le m}\{ptime(u_i, t_{j'})\}$;    (19)
```

We define

$$best(t_j) = \underset{1\le i\le m}{\mathrm{argmin}}\{ptime(u_i, t_j)\},$$

which decides the u_i that has the shortest processing time for t_j among all UAVs (line (19)). Initially (line (8)),

$$best(t_j) = \underset{1\le i\le m}{\mathrm{argmin}}\{ftime(u_i, t_j) + etime(u_i, t_j)\}.$$

The key idea of Algorithm 2 is in line (11), i.e., the next task t_j is chosen and assigned to $u_{best(t_j)}$, such that the processing time of $u_{best(t_j)}$ for t_j is the minimum among all remaining tasks. This can make $ttime(T)$ to increase slowly. Since we are only interested in the total time, the processing time $time(u_i)$ of each individual u_i really does not matter.

4.2.4. A lower bound

Theorem 2B. *A lower bound for the optimal solution of the TTM problem is*

$$Opt(I) \ge ttime^*(T) = \sum_{j=1}^{n} ptime^*(t_j).$$

Proof. The above lower bound can be justified as follows. The optimal total time cannot be shorter than the total minimum possible processing time of all tasks. □

Let I be an instance of the TTM problem. Then, we have

$$\frac{A_2(I)}{\mathrm{Opt}(I)} \le \frac{A_2(I)}{ttime^*(T)}.$$

Therefore,

$$B_2 = \boldsymbol{E}\left[\frac{A_2(I)}{ttime^*(T)}\right]$$

can be considered as an expected performance bound of Algorithm 2.

4.2.5. Performance evaluation

We use the same parameter setting as that in Section 4.1.5.

Tables 2A and 2B demonstrate our simulation results of total time minimization for both homogeneous and heterogeneous UAVs respectively. The number of tasks is $n = 10, 20, \dots, 100$. For each combination of n and τ, we generate $M = 1000$ random instances of the TTM problem. For each instance I, we apply Algorithm 2 to obtain $A_2(I)$, calculate the lower bound $ttime^*(T)$, and record the ratio $A_2(I)/ttime^*(T)$. The average of the M ratios is reported as B_2, with 99% confidence interval of ±0.14930% and ±0.32858% respectively. We have the following important observations.

- The performance bound B_2 is extremely close to one. This means that Algorithm 2 is able to efficiently find an assignment of the n tasks to the m UAVs and a flight route for each UAV, and to effectively reduce the total time.
- The performance bound B_2 is closer to one as the number of tasks increases and/or as the task execution times become longer. (Actually, for homogeneous UAVs, when n is small, B_2 may increase as n increases; however, B_2 eventually decreases as n becomes big.) This means that when there are many tasks, Algorithm 2 can more effectively and efficiently minimize the total time. Furthermore, as task execution times become longer, since the impact of route planning becomes smaller, Algorithm 2 can perform better.
- Algorithm 2 performs better for homogeneous UAVs than heterogeneous UAVs, primarily due to less randomness and variability of flight velocities and execution times.

Table 2A
Simulation results of total time minimization (Homogeneous UAVs, 99% confidence interval = ±0.14930%).

n	$\tau = 30$	$\tau = 50$	$\tau = 70$	$\tau = 90$
10	1.04732	1.03330	1.02563	1.02089
20	1.06041	1.04282	1.03323	1.02719
30	1.06588	1.04789	1.03737	1.03025
40	1.06868	1.04937	1.03866	1.03179
50	1.06951	1.05027	1.03917	1.03267
60	1.07044	1.05076	1.03968	1.03271
70	1.07060	1.05080	1.04013	1.03311
80	1.07031	1.05080	1.04016	1.03349
90	1.07029	1.05055	1.04024	1.03349
100	1.06935	1.05049	1.04033	1.03350

Table 2B
Simulation results of total time minimization (Heterogeneous UAVs, 99% confidence interval = ±0.32858%).

n	$\tau = 30$	$\tau = 50$	$\tau = 70$	$\tau = 90$
10	1.19104	1.14690	1.11796	1.09864
20	1.20495	1.15258	1.12240	1.10255
30	1.20480	1.15059	1.12022	1.10032
40	1.20197	1.14814	1.11712	1.09850
50	1.19958	1.14537	1.11457	1.09620
60	1.19567	1.14230	1.11228	1.09436
70	1.19260	1.13862	1.10992	1.09215
80	1.18927	1.13694	1.10805	1.09026
90	1.18605	1.13369	1.10603	1.08830
100	1.18297	1.13116	1.10400	1.08701

4.3. *Finished Tasks Maximization with Time Constraint (FTM-TC)*

In this section, we address the finished tasks maximization with time constraint problem.

4.3.1. *Problem definition*

The finished tasks maximization with time constraint problem is to maximize the number of finished tasks within certain time deadline.

Problem 3 (*Finished Tasks Maximization with Time Constraint*). Given a set of heterogeneous UAVs $U = \{u_1, u_2, \dots, u_m\}$, a set of tasks $T = \{t_1, t_2, \dots, t_n\}$, and a time deadline D, the ***finished tasks maximization with time constraint*** (FTM-TC) problem is to determine m disjoint subsets $T_1, T_2, \dots, T_m$, where T_i is assigned to u_i and $T_1 \cup T_2 \cup \cdots \cup T_m \subseteq T$, such that the time of u_i does not exceed D, i.e., $time(u_i) \le D$, for all $1 \le i \le m$, and the number of finished tasks, i.e.,

$$N_t = |T_1| + |T_2| + \cdots + |T_m|,$$

is maximized.

4.3.2. *NP-hardness*

Theorem 3A. *The FTM-TC problem is NP-hard.*

Proof. We provide a Turing reduction from the CTM problem to the FTM-TC problem. Let d be the longest distance:

$$d = \max\left\{ \max_{1\le i\le m, 1\le j\le n}\{dist(u_i, t_j)\}, \max_{1\le j,j'\le n}\{dist(t_j, t_{j'})\} \right\},$$

and e, f be the maximum execution time and the maximum flight time respectively:

$$e = \max_{1\le i\le m, 1\le j\le n}\{etime(u_i, t_j)\}, \quad f = d / \min_{1\le i\le m}\{vel(u_i)\},$$

and $B = n(e+f)$ be an upper bound on the completion time $ctime(T)$. It is clear that the CTM problem is basically to find the minimum D, such that all tasks can be finished by the deadline D. Such D can be found by a binary search in the range $[0, B]$. Given any algorithm A for the FTM-TC problem with time complexity $T(n)$, we can run algorithm A multiple times to find D. Initially, we set $D = B/2$. After each execution, we check whether all tasks can be finished, and adjust D accordingly. The number of times to run algorithm A is linearly proportional to the input size of B (its magnitude and precision). Therefore, we obtain an algorithm for the CTM problem with time complexity $O(($the input size of $B)T(n))$, which is a polynomial of the input size. □

4.3.3. *A heuristic algorithm*

Our heuristic algorithm to solve the FTM-TC problem is presented in Algorithm 3.

Algorithm 3: Finished Tasks Maximization with Time Constraint

Input: $U = \{u_1, u_2, ..., u_m\}$, $T = \{t_1, t_2, ..., t_n\}$, and $dist(u_i, t_j)$, $dist(t_j, t_{j'})$, $vel(u_i)$, $etime(u_i, t_j)$, for all $1 \le i \le m$ and $1 \le j, j' \le n$, and a time deadline D.
Output: $T_1, T_2, ..., T_m$, such that $time(u_i) \le D$, for all $1 \le i \le m$, and N_t is maximized.

The algorithm follows AF-Max with the following details in lines (8), (9), (13), (21), (25):

```
    $uav(t_j) \leftarrow \{i \mid ftime(u_i, t_j) + etime(u_i, t_j) \le D\}$;    (8)
    $best(t_j) \leftarrow \text{argmin}_{i\in uav(t_j)}\{ptime(u_i, t_j)\}$, if $uav(t_j) \ne \emptyset$;    (9)
    $t_j \leftarrow \text{argmin}_{t_{j'}\in T \text{ and } uav(t_{j'})\ne\emptyset}\{ptime(u_{best(t_{j'})}, t_{j'})\}$;    (13)
      if $(time(u_i) + ptime(u_i, t_{j'}) \le D)$ then    (21)
        $best(t_{j'}) \leftarrow \text{argmin}_{i\in uav(t_{j'})}\{ptime(u_i, t_{j'})\}$, if $uav(t_{j'}) \ne \emptyset$;    (25)
```

Let us define

$$uav(t_j) = \{i \mid time(u_i) + ptime(u_i, t_j) \le D\},$$

which is the set of u_i's that can still accommodate t_j within the time deadline D (line (21)). Initially (line (8)),

$$uav(t_j) = \{i \mid ftime(u_i, t_j) + etime(u_i, t_j) \le D\}.$$

The definition of $best(t_j)$ is given as

$$best(t_j) = \underset{i\in uav(t_j)}{\text{argmin}}\{ptime(u_i, t_j)\}, \quad \text{if } uav(t_j) \ne \emptyset.$$

The $best(t_j)$ value decides the u_i, such that u_i still has time to execute t_j before the deadline and has the shortest processing time for t_j among all UAVs which can execute t_j (lines (9) and (25)).

The key idea of Algorithm 3 is in line (13), i.e., the next task t_j is chosen and assigned to $u_{best(t_j)}$, such that the processing time of $u_{best(t_j)}$ for t_j is the minimum among all remaining tasks. This is to process as many tasks as possible within the time deadline.

4.3.4. *An upper bound*

Let tasks be arranged in such a way that

$$ptime^*(t_1) \le ptime^*(t_2) \le \cdots \le ptime^*(t_n).$$

Theorem 3B. *An upper bound for the optimal solution of the FTM-TC problem is*

$$Opt(I) \le N_t^* = k,$$

where k is the largest integer satisfying

$$ptime^*(t_1) + ptime^*(t_2) + \cdots + ptime^*(t_k) \le mD.$$

Proof. The above upper bound can be justified as follows. The total time of all UAVs cannot be greater than mD. To finish as many tasks as possible within the time deadline, we need to execute tasks with

Table 3A
Simulation results of finished tasks maximization with time constraint (Homogeneous UAVs, 99% confidence interval = ±0.96797%).

n	$\tau = 30$	$\tau = 50$	$\tau = 70$	$\tau = 90$
15	0.67670	0.69804	0.70533	0.71449
30	0.82486	0.84531	0.85701	0.86214
45	0.86326	0.88450	0.89473	0.90155
60	0.88153	0.90026	0.91090	0.91731
75	0.89412	0.91218	0.92109	0.92746
90	0.90186	0.91901	0.92828	0.93451
105	0.90875	0.92550	0.93400	0.93966
120	0.91228	0.92845	0.93850	0.94377
135	0.91642	0.93279	0.94174	0.94714
150	0.91932	0.93544	0.94388	0.94938

Table 3B
Simulation results of finished tasks maximization with time constraint (Heterogeneous UAVs, 99% confidence interval = ±1.03438%).

n	$\tau = 30$	$\tau = 50$	$\tau = 70$	$\tau = 90$
15	0.63610	0.69591	0.74025	0.77025
30	0.75188	0.79260	0.81743	0.83192
45	0.78364	0.81869	0.84005	0.85134
60	0.80677	0.84145	0.86108	0.87315
75	0.82155	0.85620	0.87568	0.88781
90	0.83161	0.86520	0.88400	0.89612
105	0.84021	0.87298	0.89207	0.90329
120	0.84703	0.87918	0.89723	0.90903
135	0.85284	0.88455	0.90172	0.91329
150	0.85772	0.88903	0.90582	0.91676

as short processing time as possible. According to the definition of k, it is the maximum possible number of finished tasks within the time deadline. □

Let I be an instance of the FTM-TC problem. Then, we have

$$\frac{A_3(I)}{\mathrm{Opt}(I)} \geq \frac{A_3(I)}{N_t^*}.$$

Therefore,

$$B_3 = \boldsymbol{E}\left[\frac{A_3(I)}{N_t^*}\right]$$

can be considered as an expected performance bound of Algorithm 3.

4.3.5. Performance evaluation

We use the same parameter setting as that in Section 4.1.5.

The time deadline is set as

$$D = \frac{1}{m} \times \left(\frac{2}{3}n\right) \times 0.85(10.0 + 1.5\tau),$$

such that the number of finished tasks is roughly $(2/3)n$.

Tables 3A and 3B demonstrate our simulation results of finished tasks maximization with time constraint for both homogeneous and heterogeneous UAVs respectively. The number of tasks is $n = 15, 30, \dots, 150$. For each combination of n and τ, we generate $M = 1000$ random instances of the FTM-TC problem. For each instance I, we apply Algorithm 3 to obtain $A_3(I)$, calculate the upper bound N_t^*, and record the ratio $A_3(I)/N_t^*$. The average of the M ratios is reported as B_3, with 99% confidence interval of ±0.96797% and ±1.03438% respectively. We have the following important observations.

- The performance bound B_3 is very close to one. This means that Algorithm 3 is able to efficiently find an assignment of the n tasks to the m UAVs and a flight route for each UAV, and to effectively increase the number of finished tasks.
- The performance bound B_3 is closer to one as the number of tasks increases and/or as the task execution times become longer. This means that when there are many tasks, Algorithm 3 can more effectively and efficiently maximize the number of finished tasks for a given time deadline. Furthermore, as task execution times become longer, since the impact of route planning becomes smaller, Algorithm 3 can perform better.
- Algorithm 3 performs better for homogeneous UAVs than heterogeneous UAVs, primarily due to less randomness and variability of flight velocities and execution times.

4.4. Reward Maximization with Time Constraint (RM-TC)

In this section, we address the reward maximization with time constraint problem.

4.4.1. Problem definition

The reward maximization with time constraint problem is to maximize the total reward of finished tasks within certain time deadline.

Problem 4 (*Reward Maximization with Time Constraint*)**.** Given a set of heterogeneous UAVs $U = \{u_1, u_2, \dots, u_m\}$, a set of tasks $T = \{t_1, t_2, \dots, t_n\}$, and a time deadline D, the ***reward maximization with time constraint*** (RM-TC) problem is to determine m disjoint subsets $T_1, T_2, \dots, T_m$, where T_i is assigned to u_i and $T_1 \cup T_2 \cup \cdots \cup T_m \subseteq T$, such that the time of u_i does not exceed D, i.e., $time(u_i) \leq D$, for all $1 \leq i \leq m$, and the total reward of finished tasks, i.e.,

$$R_t = \sum_{i=1}^{m} reward(u_i),$$

is maximized.

4.4.2. NP-hardness

Theorem 4A. *The RM-TC problem is NP-hard.*

Proof. If all tasks have the same reward: $reward(t_1) = reward(t_2) = \cdots = reward(t_n)$, the RM-TC problem is identical to the FTM-TC problem. In other words, the FTM-TC problem is a special case of the RM-TC problem. □

4.4.3. A heuristic algorithm

Our heuristic algorithm to solve the RM-TC problem is presented in Algorithm 4.

Algorithm 4: Reward Maximization with Time Constraint

Input: $U = \{u_1, u_2, ..., u_m\}$, $T = \{t_1, t_2, ..., t_n\}$, and $dist(u_i, t_j)$, $dist(t_j, t_{j'})$, $vel(u_i)$, $etime(u_i, t_j)$, $reward(t_j)$, for all $1 \leq i \leq m$ and $1 \leq j, j' \leq n$, and a time deadline D.
Output: $T_1, T_2, ..., T_m$, such that $time(u_i) \leq D$, for all $1 \leq i \leq m$, and R_t is maximized.

The algorithm follows AF-Max with the following details in lines (8), (9), (13), (21), (25):

```
    $uav(t_j) \leftarrow \{i \mid ftime(u_i, t_j) + etime(u_i, t_j) \leq D\}$;                    (8)
    $best(t_j) \leftarrow \mathrm{argmax}_{i \in uav(t_j)}\{reward(t_j)/ptime(u_i, t_j)\}$,
                        if $uav(t_j) \neq \emptyset$;                                           (9)
  $t_j \leftarrow \mathrm{argmax}_{t_{j'} \in T \text{ and } uav(t_{j'}) \neq \emptyset}$
                  $\{reward(t_{j'})/ptime(u_{best(t_{j'})}, t_{j'})\}$;                          (13)
      if $(time(u_i) + ptime(u_i, t_{j'}) \leq D)$ then                                        (21)
      $best(t_{j'}) \leftarrow \mathrm{argmax}_{i \in uav(t_{j'})}\{reward(t_{j'})/ptime(u_i, t_{j'})\}$,
                        if $uav(t_{j'}) \neq \emptyset$;                                        (25)
```

The definition of $uav(t_j)$ is the same as that in Section 4.3.3 (lines (8) and (21)).

The definition of $best(t_j)$ is modified as

$$best(t_j) = \operatorname*{argmax}_{i \in uav(t_j)} \left\{ \frac{reward(t_j)}{ptime(u_i, t_j)} \right\}, \quad \text{if } uav(t_j) \neq \emptyset.$$

The $best(t_j)$ value decides the u_i, such that u_i still has time to execute t_j before the deadline and has the highest reward per unit of processing time for t_j among all UAVs which can execute t_j (lines (9) and (25)).

The key idea of Algorithm 4 is in line (13), i.e., the next task t_j is chosen and assigned to $u_{best(t_j)}$, such that $u_{best(t_j)}$ has the highest reward per unit of processing time for t_j among all remaining tasks. This is to get the highest reward within the time deadline.

4.4.4. *An upper bound*

Let tasks be arranged in such a way that

$$\frac{reward(t_1)}{ptime^*(t_1)} \geq \frac{reward(t_2)}{ptime^*(t_2)} \geq \cdots \geq \frac{reward(t_n)}{ptime^*(t_n)}.$$

Theorem 4B. *An upper bound for the optimal solution of the RM-TC problem is*

$$Opt(I) \leq R_t^* = \sum_{j=1}^{k} reward(t_j) + \frac{reward(t_{k+1})}{ptime^*(t_{k+1})} \left(mD - \sum_{j=1}^{k} ptime^*(t_j) \right),$$

where k is the largest integer satisfying

$$ptime^*(t_1) + ptime^*(t_2) + \cdots + ptime^*(t_k) \leq mD.$$

Proof. The above upper bound can be justified as follows. The total time of all UAVs cannot be greater than mD. To maximize the total reward of finished tasks, we need to maximize the reward per unit of processing time. According to the definition of k, R_t^* is the maximum possible reward of finished tasks within the time deadline. □

Let I be an instance of the RM-TC problem. Then, we have

$$\frac{A_4(I)}{\mathrm{Opt}(I)} \geq \frac{A_4(I)}{R_t^*}.$$

Therefore,

$$B_4 = \boldsymbol{E}\left[\frac{A_4(I)}{R_t^*} \right]$$

can be considered as an expected performance bound of Algorithm 4.

4.4.5. *Performance evaluation*

We use the same parameter setting as that in Sections 4.1.5 and 4.3.5.

Tables 4A and 4B demonstrate our simulation results of reward maximization with time constraint for both homogeneous and heterogeneous UAVs respectively. The number of tasks is $n = 15, 30, \ldots, 150$. For each combination of n and τ, we generate $M = 1000$ random instances of the RM-TC problem. For each instance I, we apply Algorithm 4 to obtain $A_4(I)$, calculate the upper bound R_t^*, and record the ratio $A_4(I)/R_t^*$. The average of the M ratios is reported as B_4, with 99% confidence interval of ±0.76493% and ±0.88322 respectively. We have the following important observations.

- The performance bound B_4 is very close to one. This means that Algorithm 4 is able to efficiently find an assignment of the n tasks to the m UAVs and a flight route for each UAV, and to effectively increase the reward of finished tasks.
- The performance bound B_4 is closer to one as the number of tasks increases and/or as the task execution times become longer. This means that when there are many tasks, Algorithm 4 can more effectively and efficiently maximize the reward of finished tasks within a time constraint. Furthermore, as task execution times become longer, since the impact of route planning becomes smaller, Algorithm 4 can perform better.

Table 4A
Simulation results of reward maximization with time constraint (Homogeneous UAVs, 99% confidence interval = ±0.76493%).

n	$\tau = 30$	$\tau = 50$	$\tau = 70$	$\tau = 90$
15	0.71232	0.73766	0.75129	0.76104
30	0.85603	0.87823	0.89113	0.90006
45	0.88957	0.90915	0.91942	0.92730
60	0.90563	0.92312	0.93344	0.94006
75	0.91531	0.93187	0.94216	0.94769
90	0.92094	0.93752	0.94664	0.95306
105	0.92503	0.94120	0.95030	0.95614
120	0.92825	0.94455	0.95354	0.95900
135	0.93138	0.94637	0.95501	0.96101
150	0.93413	0.94832	0.95695	0.96260

Table 4B
Simulation results of reward maximization with time constraint (Heterogeneous UAVs, 99% confidence interval = ±0.88322%).

n	$\tau = 30$	$\tau = 50$	$\tau = 70$	$\tau = 90$
15	0.66862	0.71060	0.74475	0.77675
30	0.80306	0.83228	0.85493	0.87050
45	0.83836	0.86287	0.88325	0.89573
60	0.85456	0.88084	0.89658	0.90749
75	0.86555	0.88967	0.90530	0.91489
90	0.87133	0.89539	0.91051	0.92129
105	0.87821	0.90025	0.91565	0.92499
120	0.88223	0.90415	0.91874	0.92794
135	0.88578	0.90777	0.92150	0.93143
150	0.89000	0.91098	0.92444	0.93343

- Algorithm 4 performs better for homogeneous UAVs than heterogeneous UAVs, primarily due to less randomness and variability of flight velocities and execution times.

5. Distance-centric problems

In this section, we deal with distance-centric optimization problems, including the longest distance minimization problem, the total distance minimization problem, the finished tasks maximization with distance constraint problem, and the reward maximization with distance constraint problem. For each problem, we give its definition, prove its NP-hardness, present a heuristic algorithm, derive a lower bound (for a minimization problem) or an upper bound (for a maximization problem), and evaluate its performance.

5.1. *Longest Distance Minimization (LDM)*

In this section, we address the longest distance minimization problem.

5.1.1. *Problem definition*

The longest distance minimization problem is to minimize the longest distance of a set of tasks.

Problem 5 (*Longest Distance Minimization*)**.** Given a set of heterogeneous UAVs $U = \{u_1, u_2, \ldots, u_m\}$, and a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$, the *longest distance minimization* (LDM) problem is to determine a partition of T into m disjoint subsets $T_1, T_2, \ldots, T_m$, where T_i is assigned to u_i and $T_1 \cup T_2 \cup \cdots \cup T_m = T$, such that the longest distance of T, i.e.,

$$ldistance(T) = \max_{1 \leq i \leq m} \{distance(u_i)\},$$

is minimized.

5.1.2. NP-hardness

Theorem 5A. *The LDM problem is NP-hard.*

Proof. As mentioned in Section 2.2, when $vel(u_i) = 1$ and $etime(u_i, t_j) = 0$, for all $1 \le i \le m$ and $1 \le j \le n$, we have $time(u_i) = distance(u_i)$. Hence, in this special case, the LDM problem is identical to the CTM problem, which is already known to be NP-hard even when $m = 1$ using a reduction from the TSPP. □

5.1.3. A heuristic algorithm

Our heuristic algorithm to solve the LDM problem is presented in Algorithm 5.

Algorithm 5: Longest Distance Minimization

Input: $U = \{u_1, u_2, ..., u_m\}$, $T = \{t_1, t_2, ..., t_n\}$, and $dist(u_i, t_j)$, $dist(t_j, t_{j'})$, for all $1 \le i \le m$ and $1 \le j, j' \le n$.
Output: $T_1, T_2, ..., T_m$, such that $ldistance(T)$ is minimized.

The algorithm follows AF-Min with the following details in lines (8), (11), (19):

```
    $best(t_j) \leftarrow \text{argmin}_{1\le i\le m}\{dist(u_i, t_j)\}$;    (8)
    $t_j \leftarrow \text{argmin}_{t_{j'}\in T}\{distance(u_{best(t_{j'})}) + fdist(u_{best(t_{j'})}, t_{j'})\}$;    (11)
        $best(t_{j'}) \leftarrow \text{argmin}_{1\le i\le m}\{distance(u_i) + fdist(u_i, t_{j'})\}$;    (19)
```

We define

$$best(t_j) = \underset{1\le i\le m}{\text{argmin}}\{distance(u_i) + fdist(u_i, t_j)\},$$

which decides the u_i, such that if u_i processes t_j, the new flight distance of u_i is the shortest among all UAVs (line (19)). Initially (line (8)),

$$best(t_j) = \underset{1\le i\le m}{\text{argmin}}\{dist(u_i, t_j)\}.$$

The key idea of Algorithm 5 is in line (11), i.e., the next task t_j is chosen and assigned to $u_{best(t_j)}$, such that the new flight distance of $u_{best(t_j)}$ (after t_j is processed) is the minimum among all remaining tasks. This can make $ldistance(T)$ to increase slowly.

5.1.4. A lower bound

Let $dist^*(t_j)$ be the minimum flight distance to reach t_j:

$$dist^*(t_j) = \min\left\{\min_{1\le i\le m}\{dist(u_i, t_j)\}, \min_{j'\ne j}\{dist(t_{j'}, t_j)\}\right\}.$$

Theorem 5B. *A lower bound for the optimal solution of the LDM problem is*

$$\text{Opt}(I) \ge ldistance^*(T) = \frac{1}{m}\sum_{j=1}^{n} dist^*(t_j).$$

Proof. The above lower bound can be justified as follows. The optimal longest distance cannot be shorter than the total distance divided by m, while the total distance cannot be shorter than

$$\sum_{j=1}^{n} dist^*(t_j),$$

since the flight distance to reach t_j is at least $dist^*(t_j)$. □

Let I be an instance of the LDM problem. Then, we have

$$\frac{A_5(I)}{\text{Opt}(I)} \le \frac{A_5(I)}{ldistance^*(T)}.$$

Table 5
Simulation results of longest distance minimization (99% confidence interval = ±1.70573%).

n	$m = 3$	$m = 5$	$m = 7$	$m = 9$
15	1.83001	2.15761	2.46208	2.76499
30	1.76137	1.93912	2.16648	2.34352
45	1.73176	1.86453	2.00664	2.16382
60	1.70407	1.82211	1.92672	2.06167
75	1.67991	1.79521	1.87785	2.00090
90	1.66439	1.76451	1.85102	1.94942
105	1.65486	1.74373	1.82498	1.91043
120	1.63568	1.72506	1.80488	1.87775
135	1.63284	1.70645	1.78087	1.85071
150	1.62526	1.69886	1.76536	1.83470

Therefore,

$$B_5 = E\left[\frac{A_5(I)}{ldistance^*(T)}\right]$$

can be considered as an expected performance bound of Algorithm 5.

5.1.5. Performance evaluation

We use the same parameter setting as that in Section 4.1.5.

Table 5 demonstrates our simulation results of longest distance minimization. Since flight velocities and execution speeds are not considered, we do not distinguish homogeneous and heterogeneous UAVs. The number of tasks is $n = 15, 30, \ldots, 150$. The number of UAVs is $m = 3, 5, 7, 9$. For each combination of n and m, we generate $M = 1000$ random instances of the LDM problem. For each instance I, we apply Algorithm 5 to obtain $A_5(I)$, calculate the lower bound $ldistance^*(T)$, and record the ratio $A_5(I)/ldistance^*(T)$. The average of the M ratios is reported as B_5, with 99% confidence interval of ±1.70573%. We have the following important observations.

- The performance bound B_5 is moderately close to one. Further improvement is possible by tightening the lower bound and finding a more effective heuristic algorithm.
- The performance bound B_5 increases as the number of UAVs increases. This means that it is more difficult to manage more UAVs in reducing the longest distance.
- The performance bound B_5 decreases as the number of tasks increases. This means that when there are many tasks, Algorithm 5 can more effectively and efficiently minimize the longest distance.

5.2. Total Distance Minimization (TDM)

In this section, we address the total distance minimization problem.

5.2.1. Problem definition

The total distance minimization problem is to minimize the total distance of a set of tasks.

Problem 6 (*Total Distance Minimization*)**.** Given a set of heterogeneous UAVs $U = \{u_1, u_2, \ldots, u_m\}$, and a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$, the *total distance minimization* (TDM) problem is to determine a partition of T into m disjoint subsets $T_1, T_2, \ldots, T_m$, where T_i is assigned to u_i and $T_1 \cup T_2 \cup \cdots \cup T_m = T$, such that the total distance of T, i.e.,

$$tdistance(T) = \sum_{i=1}^{m} distance(u_i),$$

is minimized.

5.2.2. NP-hardness

Theorem 6A. *The TDM problem is NP-hard.*

Proof. When $m = 1$, the TDM problem is identical to the LDM problem, which is already known to be NP-hard when $m = 1$ using a reduction from the TSPP. □

5.2.3. A heuristic algorithm

Our heuristic algorithm to solve the TDM problem is presented in Algorithm 6.

```
Algorithm 6: Total Distance Minimization

Input: $U = \{u_1, u_2, ..., u_m\}$, $T = \{t_1, t_2, ..., t_n\}$, and $dist(u_i, t_j)$, $dist(t_j, t_{j'})$, for all $1 \le i \le m$ and $1 \le j, j' \le n$.
Output: $T_1, T_2, ..., T_m$, such that $tdistance(T)$ is minimized.

The algorithm follows AF-Min with the following details in lines (8), (11), (19):
    $best(t_j) \leftarrow \mathrm{argmin}_{1 \le i \le m}\{dist(u_i, t_j)\}$;    (8)
  $t_j \leftarrow \mathrm{argmin}_{t_{j'} \in T}\{fdist(u_{best(t_{j'})}, t_{j'})\}$;    (11)
      $best(t_{j'}) \leftarrow \mathrm{argmin}_{1 \le i \le m}\{fdist(u_i, t_{j'})\}$;    (19)
```

We define

$$best(t_j) = \underset{1 \le i \le m}{\mathrm{argmin}}\{fdist(u_i, t_j)\},$$

which decides the u_i that has the shortest flight distance to t_j among all UAVs (line (19)). Initially (line (8)),

$$best(t_j) = \underset{1 \le i \le m}{\mathrm{argmin}}\{dist(u_i, t_j)\}.$$

The key idea of Algorithm 6 is in line (11), i.e., the next task t_j is chosen and assigned to $u_{best(t_j)}$, such that the flight distance from $u_{best(t_j)}$ to t_j is the minimum among all remaining tasks. This can make $tdistance(T)$ to increase slowly. Since we are only interested in the total distance, the flight distance $distance(u_i)$ of each individual u_i really does not matter.

5.2.4. A lower bound

Theorem 6B. *A lower bound for the optimal solution of the TDM problem is*

$$Opt(I) \ge tdistance^*(T) = \sum_{j=1}^{n} dist^*(t_j).$$

Proof. The above lower bound can be justified as follows. The optimal total distance cannot be shorter than the total minimum distance to reach tasks. □

Let I be an instance of the TDM problem. Then, we have

$$\frac{A_6(I)}{\mathrm{Opt}(I)} \le \frac{A_6(I)}{tdistance^*(T)}.$$

Therefore,

$$B_6 = \boldsymbol{E}\left[\frac{A_2(I)}{tdistance^*(T)}\right]$$

can be considered as an expected performance bound of Algorithm 6.

5.2.5. Performance evaluation

We use the same parameter setting as that in Section 4.1.5.

Table 6 demonstrates our simulation results of total distance minimization. Since flight velocities and execution speeds are not considered, we do not distinguish homogeneous and heterogeneous UAVs. The number of tasks is $n = 15, 30, \ldots, 150$. The number of UAVs is $m = 3, 5, 7, 9$. For each combination of n and m, we generate $M = 1000$ random instances of the TDM problem. For each instance I, we apply Algorithm 6 to obtain $A_6(I)$, calculate the lower bound $tdistance^*(T)$, and record the ratio $A_6(I)/tdistance^*(T)$. The average of the M ratios is reported as B_6, with 99% confidence interval of ±0.71157%. We have the following important observations.

Table 6
Simulation results of total distance minimization (99% confidence interval = ±0.71157%).

n	$m = 3$	$m = 5$	$m = 7$	$m = 9$
15	1.34176	1.26310	1.21041	1.18239
30	1.39468	1.32278	1.27797	1.24838
45	1.41577	1.35765	1.31882	1.28146
60	1.42891	1.37747	1.33921	1.30910
75	1.43934	1.38952	1.35535	1.32930
90	1.44793	1.39915	1.36759	1.34036
105	1.44796	1.40579	1.37606	1.35304
120	1.45360	1.41375	1.38404	1.36153
135	1.45393	1.41833	1.38969	1.36821
150	1.45616	1.41968	1.39334	1.37425

- The performance bound B_6 is reasonably close to one. Further improvement is possible by tightening the lower bound and finding a more effective heuristic algorithm.
- The performance bound B_6 decreases as the number of UAVs increases. This means that Algorithm 6 is more effective and efficient in managing more UAVs to reduce the total distance.
- The performance bound B_6 increases as the number of tasks increases. This means that it is more difficult to handle more tasks in reducing the total distance.

5.3. Finished Tasks Maximization with Distance Constraint (FTM-DC)

In this section, we address the finished tasks maximization with distance constraint problem.

5.3.1. Problem definition

The finished tasks maximization with distance constraint problem is to maximize the number of finished tasks within certain distance limitation.

Problem 7 (*Finished Tasks Maximization with Distance Constraint*)**.** Given a set of heterogeneous UAVs $U = \{u_1, u_2, \ldots, u_m\}$, a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$, and a distance limitation $maxdist(u_i)$ for each u_i, the *finished tasks maximization with distance constraint* (FTM-DC) problem is to determine m disjoint subsets $T_1, T_2, \ldots, T_m$, where T_i is assigned to u_i and $T_1 \cup T_2 \cup \cdots \cup T_m \subseteq T$, such that the flight distance of u_i does not exceed $maxdist(u_i)$, i.e., $distance(u_i) \le maxdist(u_i)$, for all $1 \le i \le m$, and the number of finished tasks, i.e.,

$$N_d = |T_1| + |T_2| + \cdots + |T_m|,$$

is maximized.

5.3.2. NP-hardness

Theorem 7A. *The FTM-DC problem is NP-hard.*

Proof. We provide a Turing reduction from the LDM problem to the FTM-DC problem. Let d be the longest distance defined in the proof of Theorem 3A, and $B = nd$ be an upper bound on the longest distance $ldistance(T)$. Assume that $maxdist(u_i) = D$ for all $1 \le i \le m$. It is clear that the LDM problem is basically to find the minimum D, such that all tasks can be finished within the distance limitation D. Such D can be found by a binary search in the range $[0, B]$. Given any algorithm A for the FTM-DC problem with time complexity $T(n)$, we can run algorithm A multiple times to find D. The number of times to run algorithm A is linearly proportional to the input size of B (its magnitude and precision). Therefore, we obtain an algorithm for the LDM problem with time complexity O((the input size of B)$T(n)$), which is a polynomial of the input size. □

Algorithm 7: Finished Tasks Maximization with Distance Constraint

Input: $U = \{u_1, u_2, ..., u_m\}$, $T = \{t_1, t_2, ..., t_n\}$, and $dist(u_i, t_j)$, $dist(t_j, t_{j'})$, $maxdist(u_i)$, for all $1 \le i \le m$ and $1 \le j, j' \le n$.
Output: $T_1, T_2, ..., T_m$, such that $distance(u_i) \le maxdist(u_i)$, for all $1 \le i \le m$, and N_d is maximized.

The algorithm follows AF-Max with the following details in lines (8), (9), (13), (21), (25):

```
    $uav(t_j) \leftarrow \{i \mid dist(u_i, t_j) \le maxdist(u_i)\}$;    (8)
    $best(t_j) \leftarrow \text{argmin}_{i\in uav(t_j)}\{fdist(u_i, t_j)\}$, if $uav(t_j) \ne \emptyset$;    (9)
    $t_j \leftarrow \text{argmin}_{t_{j'}\in T \text{ and } uav(t_{j'})\ne\emptyset}\{fdist(u_{best(t_{j'})}, t_{j'})\}$;    (13)
      if $(distance(u_i) + fdist(u_i, t_{j'}) \le maxdist(u_i))$ then    (21)
      $best(t_{j'}) \leftarrow \text{argmin}_{i\in uav(t_{j'})}\{fdist(u_i, t_{j'})\}$, if $uav(t_{j'}) \ne \emptyset$;    (25)
```

5.3.3. A heuristic algorithm

Our heuristic algorithm to solve the FTM-DC problem is presented in Algorithm 7.

Let us define

$$uav(t_j) = \{i \mid distance(u_i) + fdist(u_i, t_j) \le maxdist(u_i)\},$$

which is the set of u_i's that can still accommodate t_j within the distance limitation $maxdist(u_i)$ (line (21)). Initially (line (8)),

$$uav(t_j) = \{i \mid dist(u_i, t_j) \le maxdist(u_i)\}.$$

The definition of $best(t_j)$ is given as

$$best(t_j) = \underset{i\in uav(t_j)}{\text{argmin}}\{fdist(u_i, t_j)\}, \quad \text{if } uav(t_j) \ne \emptyset.$$

The $best(t_j)$ value decides the u_i, such that u_i still has distance to execute t_j before the limit and has the shortest flight distance to t_j among all UAVs which can execute t_j (lines (9) and (25)).

The key idea of Algorithm 7 is in line (13), i.e., the next task t_j is chosen and assigned to $u_{best(t_j)}$, such that the flight distance from $u_{best(t_j)}$ to t_j is the minimum among all remaining tasks. This is to process as many tasks as possible within the distance limitation.

5.3.4. An upper bound

Let tasks be arranged in such a way that

$$dist^*(t_1) \le dist^*(t_2) \le \cdots \le dist^*(t_n).$$

Theorem 7B. *An upper bound for the optimal solution of the FTM-DC problem is*

$$Opt(I) \le N_d^* = k,$$

where k is the largest integer satisfying

$$dist^*(t_1) + dist^*(t_2) + \cdots + dist^*(t_k) \le \sum_{i=1}^{m} maxdist(u_i).$$

Proof. The above upper bound can be justified as follows. The total distance of all UAVs cannot be greater than

$$\sum_{i=1}^{m} maxdist(u_i).$$

To finish as many tasks as possible within the distance limitation, we need to execute tasks with as short flight distance as possible. According to the definition of k, it is the maximum possible number of finished tasks within the distance limitation. □

Table 7
Simulation results of finished tasks maximization with distance constraint (99% confidence interval = ±2.07925%).

n	$m = 3$	$m = 5$	$m = 7$	$m = 9$
15	0.47986	0.50554	0.55025	0.62380
30	0.54513	0.57617	0.61219	0.68270
45	0.56640	0.60623	0.64165	0.72209
60	0.58357	0.62614	0.66584	0.74911
75	0.59626	0.63837	0.68421	0.77829
90	0.60258	0.65440	0.70408	0.80321
105	0.61041	0.66042	0.71631	0.82365
120	0.61858	0.67084	0.73940	0.84820
135	0.62481	0.67861	0.75387	0.86660
150	0.62910	0.68571	0.77121	0.88617

Let I be an instance of the FTM-DC problem. Then, we have

$$\frac{A_7(I)}{\text{Opt}(I)} \ge \frac{A_7(I)}{N_d^*}.$$

Therefore,

$$B_7 = \boldsymbol{E}\left[\frac{A_7(I)}{N_d^*}\right]$$

can be considered as an expected performance bound of Algorithm 7.

5.3.5. Performance evaluation

We use the same parameter setting as that in Section 4.1.5.

Task distance limitations are randomly, uniformly, and independently distributed in the range $0.8D \le maxdist(u_i) < 1.2D$, where $D = 7n + 150$.

Table 7 demonstrates our simulation results of finished tasks maximization with distance constraint. Since flight velocities and execution speeds are not considered, we do not distinguish homogeneous and heterogeneous UAVs. The number of tasks is $n = 15, 30, \ldots, 150$. The number of UAVs is $m = 3, 5, 7, 9$. For each combination of n and τ, we generate $M = 1000$ random instances of the FTM-DC problem. For each instance I, we apply Algorithm 7 to obtain $A_7(I)$, calculate the upper bound N_d^*, and record the ratio $A_7(I)/N_d^*$. The average of the M ratios is reported as B_7, with 99% confidence interval of ±2.07925%. We have the following important observations.

- The performance bound B_7 is moderately close to one. Further improvement is possible by tightening the upper bound and finding a more effective heuristic algorithm.
- The performance bound B_7 increases as the number of UAVs increases. This means that Algorithm 7 is more effective and efficient in managing more UAVs to maximize the number of finished tasks within given distance limitation.
- The performance bound B_7 increases as the number of tasks increases. This means that when there are many tasks, Algorithm 7 can more effectively and efficiently maximize the number of finished tasks.

5.4. Reward Maximization with Distance Constraint (RM-DC)

In this section, we address the reward maximization with distance constraint problem.

5.4.1. Problem definition

The reward maximization with distance constraint problem is to maximize the total reward of finished tasks within certain distance limitation.

Problem 8 (*Reward Maximization with Distance Constraint*)**.** Given a set of heterogeneous UAVs $U = \{u_1, u_2, \ldots, u_m\}$, a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$, and a distance limitation $maxdist(u_i)$ for each u_i, the *reward maximization with distance constraint* (RM-DC) problem is to determine m disjoint subsets $T_1, T_2, \ldots, T_m$, where T_i is assigned to u_i and $T_1 \cup T_2 \cup \cdots \cup T_m \subseteq T$, such that the flight distance of u_i does not exceed $maxdist(u_i)$, i.e., $distance(u_i) \leq maxdist(u_i)$, for all $1 \leq i \leq m$, and the total reward of finished tasks, i.e.,

$$R_d = \sum_{i=1}^{m} reward(u_i),$$

is maximized.

5.4.2. NP-hardness

Theorem 8A. *The RM-DC problem is NP-hard.*

Proof. If all tasks have the same reward: $reward(t_1) = reward(t_2) = \cdots = reward(t_n)$, the RM-DC problem is identical to the FTM-DC problem. In other words, the FTM-DC problem is a special case of the RM-DC problem. □

5.4.3. A heuristic algorithm

Our heuristic algorithm to solve the RM-DC problem is presented in Algorithm 8.

The definition of $uav(t_j)$ is the same as that in Section 5.3.3 (lines (8) and (21)).

The definition of $best(t_j)$ is modified as

$$best(t_j) = \operatorname*{argmax}_{i \in uav(t_j)} \left\{ \frac{reward(t_j)}{fdist(u_i, t_j)} \right\}, \quad \text{if } uav(t_j) \neq \emptyset.$$

The $best(t_j)$ value decides the u_i, such that u_i still has distance to execute t_j before the limit and has the highest reward per unit of flight distance for t_j among all UAVs which can execute t_j (lines (9) and (25)).

Algorithm 8: Reward Maximization with Distance Constraint

Input: $U = \{u_1, u_2, ..., u_m\}$, $T = \{t_1, t_2, ..., t_n\}$, and $dist(u_i, t_j)$, $dist(t_j, t_{j'})$, $maxdist(u_i)$, $reward(t_j)$, for all $1 \leq i \leq m$ and $1 \leq j, j' \leq n$.
Output: $T_1, T_2, ..., T_m$, such that $distance(u_i) \leq maxdist(u_i)$, for all $1 \leq i \leq m$, and R_d is maximized.

```
The algorithm follows AF-Max with the following details in lines (8), (9), (13), (21), (25):
    $uav(t_j) \leftarrow \{i \mid dist(u_i, t_j) \leq maxdist(u_i)\}$;                          (8)
    $best(t_j) \leftarrow \mathrm{argmax}_{i \in uav(t_j)}\{reward(t_j)/fdist(u_i, t_j)\}$,
                                    if $uav(t_j) \neq \emptyset$;                                 (9)
    $t_j \leftarrow \mathrm{argmax}_{t_{j'} \in T \text{ and } uav(t_{j'}) \neq \emptyset}$
                          $\{reward(t_{j'})/fdist(u_{best(t_{j'})}, t_{j'})\}$;                  (13)
      if $(distance(u_i) + fdist(u_i, t_{j'}) \leq maxdist(u_i))$ then                          (21)
      $best(t_{j'}) \leftarrow \mathrm{argmax}_{i \in uav(t_{j'})}\{reward(t_{j'})/fdist(u_i, t_{j'})\}$,
                                    if $uav(t_{j'}) \neq \emptyset$;                              (25)
```

The key idea of Algorithm 8 is in line (13), i.e., the next task t_j is chosen and assigned to $u_{best(t_j)}$, such that $u_{best(t_j)}$ has the highest reward per unit of flight distance for t_j among all remaining tasks. This is to get the highest reward within the distance limitation.

5.4.4. An upper bound

Let tasks be arranged in such a way that

$$\frac{reward(t_1)}{dist^*(t_1)} \geq \frac{reward(t_2)}{dist^*(t_2)} \geq \cdots \geq \frac{reward(t_n)}{dist^*(t_n)}.$$

Theorem 8B. *An upper bound for the optimal solution of the RM-DC problem is*

$$Opt(I) \leq R_d^* = \sum_{j=1}^{k} reward(t_j)$$

$$+ \frac{reward(t_{k+1})}{dist^*(t_{k+1})} \left(\sum_{i=1}^{m} maxdist(u_i) - \sum_{j=1}^{k} dist^*(t_j) \right),$$

where k is the largest integer satisfying

$$dist^*(t_1) + dist^*(t_2) + \cdots + dist^*(t_k) \leq \sum_{i=1}^{m} maxdist(u_i).$$

Proof. The above upper bound can be justified as follows. The total distance of all UAVs cannot be greater than

$$\sum_{i=1}^{m} maxdist(u_i).$$

To maximize the total reward of finished tasks, we need to maximize the reward per unit of flight distance. According to the definition of k, R_d^* is the maximum possible reward of finished tasks within the distance limitation. □

Let I be an instance of the RM-DC problem. Then, we have

$$\frac{A_8(I)}{\mathrm{Opt}(I)} \geq \frac{A_8(I)}{R_d^*}.$$

Therefore,

$$B_8 = \boldsymbol{E}\left[\frac{A_8(I)}{R_d^*}\right]$$

can be considered as an expected performance bound of Algorithm 8.

5.4.5. Performance evaluation

We use the same parameter setting as that in Sections 4.1.5 and 5.3.5.

Table 8 demonstrates our simulation results of reward maximization with distance constraint. Since flight velocities and execution speeds are not considered, we do not distinguish homogeneous and heterogeneous UAVs. The number of tasks is $n = 15, 30, \ldots, 150$. The number of UAVs is $m = 3, 5, 7, 9$. For each combination of n and τ, we generate $M = 1000$ random instances of the RM-DC problem. For each instance I, we apply Algorithm 8 to obtain $A_8(I)$, calculate the upper bound R_d^*, and record the ratio $A_8(I)/R_d^*$. The average of the M ratios is reported as B_8, with 99% confidence interval of ±2.17163%. We have the following important observations.

- The performance bound B_8 is moderately close to one. Further improvement is possible by tightening the upper bound and finding a more effective heuristic algorithm.
- The performance bound B_8 increases as the number of UAVs increases. This means that Algorithm 8 is more effective and efficient in managing more UAVs to maximize the reward of finished tasks within given distance limitation.
- The performance bound B_8 increases as the number of tasks increases. This means that when there are many tasks, Algorithm 8 can more effectively and efficiently maximize the reward of finished tasks.

Table 8
Simulation results of reward maximization with distance constraint (99% confidence interval = ±2.17163%).

n	$m = 3$	$m = 5$	$m = 7$	$m = 9$
15	0.45966	0.51446	0.58855	0.67071
30	0.51881	0.57561	0.64769	0.73373
45	0.54231	0.60765	0.68296	0.76891
60	0.55706	0.62768	0.70289	0.79266
75	0.56849	0.64233	0.72570	0.81550
90	0.58355	0.65304	0.74287	0.83600
105	0.58845	0.66675	0.75697	0.85129
120	0.59553	0.67802	0.77154	0.86597
135	0.60281	0.68770	0.78707	0.87921
150	0.60854	0.69536	0.79796	0.89105

6. Related work

In this section, we review related research, which is briefly divided into two categories, i.e., task assignment and flight planning.

6.1. Task assignment

Several researchers have investigated task allocation, task assignment, and task scheduling on multiple UAVs.

Cui et al. proposed a distributed task allocation algorithm for heterogeneous UAVs to minimize the sum of task completion times with constraints on execution capability, time window, and fuel consumption [14]. Hu and Yang presented a decentralized auction algorithm for multiple farming task assignment on heterogeneous agricultural UAVs to maximize the total reward of task execution with a total resource consumption constraint [6]. Mao et al. took a double-layer deep reinforcement learning approach to task scheduling, so that homogeneous UAVs can maximize the total number of executed tasks with a total flight distance constraint [15]. Schumacher et al. gave a mixed integer linear program formulation for task assignment on homogeneous UAVs to minimize the total flight time, where tasks have timing and precedence constraints [2]. Yi et al. addressed task allocation on homogeneous UAVs to minimize the total flying distance between drones and tasks, where each drone has certain maximum number of tasks it can carry [16]. Zhang and Chen devised a clone selection algorithm for task allocation on homogeneous UAVs, to maximize the number of assigned tasks and the benefit of performing tasks, and to minimize resource cost and time consumption, simultaneously [17].

Various other approaches to task allocation and task assignment on multiple UAVs have also been investigated, e.g., leader–follower coalition [18], distributed task inclusion and dynamic grouping allocation [19], human-agent collaboration [20], negotiation [21], team-based approach [22], quantum genetic algorithm [23].

6.2. Flight planning

Several researchers have studied path planning, trajectory planning, mission planning, and route optimization for a single UAV and multiple UAVs.

Bertuccelli et al. considered decentralized task assignment on heterogeneous UAVs to maximize the total path-dependent reward, with extensions to obstacle region avoidance and sensing noise minimization [24]. Fu et al. used an auction algorithm and a consensus algorithm to find task sequences and paths for UAVs with certain resources, so as to maximize the reward of finished tasks with resource requirements [1]. Geng et al. employed the particle swarm optimization method to plan rescue routes, so as to maximize the number of survivors who have limited and different survival times [25]. Ozkan solved the distance-constrained multi-based multi-UAV routing problem by integrating simulated annealing and local search metaheuristics with an integer linear programming model [7]. Sullivan et al. minimized the energy and time required to complete all tasks in multi-robot routing by using auction bidding and resolution algorithms [26]. Wang et al. described several path planning algorithms based on distributed particle swarm optimization for UAV swarms conducting a reconnaissance mission [3]. Yao and Ansari constructed an online algorithm for a single drone to minimize its journey time (including transition time, transmission time, and processing time) in traveling through all locations of interest by fog node and flying speed optimization, with an energy consumption constraint (including energy for wireless transmission, propulsion, and hovering) and task completion deadlines [27]. Yin et al. designed a deep migration reinforcement learning algorithm to find a distribution scheme (i.e., a mission plan) for a swarm of heterogeneous UAVs to transport rescue materials to rescue areas with different urgency and needs [9]. Zhou et al. developed a bat algorithm to find an accident-free, short, and safe flight path for a single UAV in a complex three-dimensional battlefield environment [28].

6.3. Comments on existing research

There has been none existing paper which studies UAV task scheduling within the traditional framework of combinatorial optimization. For instance, none existing studies have put makespan as the main optimization objective. Furthermore, none existing studies are able to compare the solutions of their algorithms with optimal solutions. This paper is to make effort in this direction, which is the first paper studying task scheduling on heterogeneous UAVs using a combinatorial optimization approach.

7. Concluding remarks

We have proposed and solved several combinatorial optimization problems for task scheduling on heterogeneous UAVs. These problems aim to optimize various performance, cost, and other important considerations, such as the maximum task processing time, the total task processing time, the maximum flight distance, the total flight distance, the number of finished tasks, and the total reward of finished tasks. We have proved that all these problems are NP-hard using reductions from the classic traveling salesman problem. We have developed efficient and effective heuristic algorithms to solve these problems. All our algorithms fit into two algorithmic frameworks, which can serve as algorithmic templates for further heuristic algorithms to solve many other problems. A unique feature of our study is to compare the performance of our heuristic algorithms with optimal solutions. In doing so, we have derived lower/upper bounds for the optimal solutions. Our simulation results show that our heuristic algorithms are able to produce near-optimal solutions. To the best of our knowledge, there has been no similar study in the existing literature.

We would like to mention two possible directions for future research. (1) First, there is still room for improving the performance bounds of our algorithms, especially for distance-centric optimization problems. There are two possible ways for performance bound improvement. The first way is to design more effective heuristic algorithms, which are likely to be more sophisticated than the greedy algorithms in this paper. The second way is to discover tighter lower or upper bounds, which need deeper insights to find. (2) Second, we may consider task and mission scheduling for heterogeneous UAVs with multiple (e.g., completion time, flight distance, and resource consumption) constraints. It is conceivable that such optimization problems are more challenging and require harder investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The author sincerely appreciates the anonymous reviewers for their comments and suggestions on improving the manuscript.

Appendix. Notations and definitions

Notation	Definition
U	A set of heterogeneous/homogeneous UAVs
u_i	A UAV
T	A set of tasks
t_j	A task
$pos(u_i)$	$= (x(u_i), y(u_i), z(u_i))$, the position (i.e., the initial location) of u_i
$pos(t_j)$	$= (x(t_j), y(t_j), z(t_j))$, the position of t_j
$dist(u_i, t_j)$	The distance between $pos(u_i)$ and $pos(t_j)$
$dist(t_j, t_{j'})$	The distance between $pos(t_j)$ and $pos(t_{j'})$
$vel(u_i)$	The flight velocity of u_i
vel	The flight velocity of homogeneous UAVs
$ftime(u_i, t_j)$	$= dist(u_i, t_j)/vel(u_i)$, the flight time of u_i from $pos(u_i)$ to $pos(t_j)$
$ftime_i(t_j, t_{j'})$	$= dist(t_j, t_{j'})/vel(u_i)$, the flight time of u_i from $pos(t_j)$ to $pos(t_{j'})$
$etime(u_i, t_j)$	The execution time of u_i to process t_j
$etime(t_j)$	The execution time of t_j for homogeneous UAVs
$reward(t_j)$	The reward to finish t_j
T_i	$= \{t_{j_1}, t_{j_2}, \dots, t_{j_{n_i}}\}$, a set of tasks assigned to u_i
T_i	$= (t_{j_1}, t_{j_2}, \dots, t_{j_{n_i}})$, a sequence of tasks assigned to u_i (i.e., a flight route of u_i)
$(T_1, T_2, \dots, T_m)$	A schedule of T on the m UAVs
$ptime(t_j)$	The processing time of t_j
$time(u_i)$	The time for u_i to process tasks in T_i (i.e., the total processing time)
$distance(u_i)$	The flight distance of u_i
$ctime(T)$	$= \max\{time(u_1), time(u_2), \dots, time(u_m)\}$, the completion time of T
$ttime(T)$	$= time(u_1) + time(u_2) + \cdots + time(u_m)$, the total time of T
$ldistance(T)$	$= \max\{distance(u_1), distance(u_2), \dots, distance(u_m)\}$, the longest distance of T
$tdistance(T)$	$= distance(u_1) + distance(u_2) + \cdots + distance(u_m)$, the total distance of T
$reward(u_i)$	The reward of u_i, i.e., the total reward of tasks in T_i
G	A weighted directed graph
P_i	$= (u_i, t_{j_1}, t_{j_2}, \dots, t_{j_{n_i}})$, a path
$length(P_i)$	The length of P_i
L	$= (v_{j_1}, v_{j_2}, \dots, v_{j_n}, v_{j_1})$, a traveling salesman loop
$length(L)$	The length of L
I	An instance
$A(I)$	The solution of a heuristic algorithm A for I
$\mathrm{Opt}(I)$	The optimal solution of I
B	A performance bound or an expected performance bound
$loc(u_i)$	$\in \{0, 1, 2, \dots, n\}$, the current location of u_i
$ptime(u_i, t_j)$	The processing time of u_i for t_j
$fdist(u_i, t_j)$	The flight distance from u_i to t_j
$best(t_j)$	The best u_i that processes t_j
$uav(t_j)$	The set of u_i's that can still accommodate t_j within its performance or resource constraint
$ptime^*(t_j)$	The minimum possible processing time of t_j
$ctime^*(T)$	A lower bound for the optimal solution of the CTM problem
$ttime^*(T)$	A lower bound for the optimal solution of the TTM problem
D	Time deadline
N_t, N_d	The number of finished tasks
N_t^*, N_d^*	An upper bound for the optimal solution of the FTM-TC and FTM-DC problems
R_t, R_d	The total reward of finished tasks
R_t^*, R_d^*	An upper bound for the optimal solution of the RM-TC and RM-DC problems
$dist^*(t_j)$	The minimum flight distance to reach t_j
$ldistance^*(T)$	A lower bound for the optimal solution of the LDM problem
$tdistance^*(T)$	A lower bound for the optimal solution of the TDM problem
$maxdist(u_i)$	distance limitation for u_i
B_r	An expected performance bound of Algorithm r, $1 \le r \le 8$

References

[1] X. Fu, P. Feng, X. Gao, Swarm UAVs task and resource dynamic assignment algorithm based on task sequence mechanism, IEEE Access 7 (2019) 41090–41100.

[2] C. Schumacher, P. Chandler, M. Pachter, UAV Task Assignment with Timing Constraints, AFRL-VA-WP-TP-2003-315, Defense Technical Information Center, 2003.

[3] Y. Wang, P. Bai, X. Liang, W. Wang, J. Zhang, Q. Fu, Reconnaissance mission conducted by UAV swarms based on distributed PSO path planning algorithms, IEEE Access 7 (2019) 105086–105099.

[4] A. Alhaqbani, H. Kurdi, K. Youcef-Toumi, Fish-inspired task allocation algorithm for multiple unmanned aerial vehicles in search and rescue missions, Remote Sens. 13 (1) (2021) 27, http://dx.doi.org/10.3390/rs13010027.

[5] F. Aljalaud, H.A. Kurdi, Autonomous task allocation for multi-UAV systems based on area-restricted search behavior in animals, Procedia Comput. Sci. 191 (2021) 246–253.

[6] J. Hu, J. Yang, Application of distributed auction to multi-UAV task assignment in agriculture, Int. J. Precis. Agric. Aviat. 1 (1) (2018) 44–50.

[7] O. Ozkan, Optimization of the distance-constrained multi-based multi-UAV routing problem with simulated annealing and local search-based metaheuristic to detect forest fires: The case of Turkey, Appl. Soft Comput. 113 (Part B) (2021) 108015.

[8] M. Yan, H. Yuan, J. Xu, Y. Yu, L. Jin, Task allocation and route planning of multiple UAVs in a marine environment based on an improved particle swarm optimization algorithm, EURASIP J. Adv. Signal Process. (2021) 94, http://dx.doi.org/10.1186/s13634-021-00804-9.

[9] Y. Yin, Y. Guo, Q. Su, Z. Wang, Task allocation of multiple unmanned aerial vehicles based on deep transfer reinforcement learning, Drones 6 (8) (2022) 215, http://dx.doi.org/10.3390/drones6080215.

[10] J. Bellingham, M. Tillerson, A. Richards, J.P. How, Multi-Task Allocation and Path Planning for Cooperating UAVs, Cooperative Control: Models, Applications and Algorithms, Springer, Boston, MA, 2003, pp. 23–41,

[11] Q. Peng, H. Wu, R. Xue, Review of dynamic task allocation methods for UAV swarms oriented to ground targets, Complex Syst. Model. Simul. 1 (3) (2021) 163–175.

[12] Y.B. Sebbane, Multi-UAV Planning and Task Allocation, CRC Press, Boca Raton, FL, 2021.

[13] M.R. Garey, D.S. Johnson, Computers and Intractability – a Guide to the Theory of NP-Completeness, W. H. Freeman, New York, 1979.

[14] W. Cui, R. Li, Y. Feng, Y. Yang, Distributed task allocation for a multi-UAV system with time window constraints, Drones 6 (9) (2022) 226, http://dx.doi.org/10.3390/drones6090226.

[15] X. Mao, G. Wu, M. Fan, DL-DRL: A double-layer deep reinforcement learning approach for large-scale task scheduling of multi-UAV, 2022, arXiv:2208.02447, 21 Aug. https://arxiv.org/abs/2208.02447.

[16] B. Yi, J. Lv, X. Wang, J. Chen, K. Li, Digital twin constructed spatial structure for flexible and efficient task allocation of drones in mobile networks, IEEE J. Sel. Areas Commun. (2023) submitted for publication.

[17] X. Zhang, X. Chen, UAV task allocation based on clone selection algorithm, Wirel. Commun. Mob. Comput. 2021 (2021) 5518927.

[18] J. Chen, D. Sun, Resource constrained multirobot task allocation based on leader-follower coalition methodology, Int. J. Robot. Res. 30 (12) (2011) 1423–1434.

[19] X. Chen, P. Zhang, G. Du, F. Li, A distributed method for dynamic multi-robot task allocation problems with critical time constraints, Robot. Auton. Syst. 118 (2019) 31–46.

[20] S.D. Ramchurn, J.E. Fischer, Y. Ikuno, F. Wu, J. Flann, A. Waldock, A study of human-agent collaboration for multi-UAV task allocation in dynamic environments, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015, pp. 1184–1192.
[21] P.B. Sujit, A. Sinha, D. Ghose, Multiple UAV task allocation using negotiation, in: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, 2006, pp. 471–478.
[22] T.K. Venugopalan, K. Subramanian, S. Suresh, Multi-UAV task allocation: A team-based approach, in: IEEE Symposium Series on Computational Intelligence, 2015, pp. 45–50.
[23] Z. Wang, X. Yan, Multi-UAV task assignment based on quantum genetic algorithm, J. Phys.: Conf. Ser. 1824 (2021) 012010, http://dx.doi.org/10.1088/1742-6596/1824/1/012010.
[24] L. Bertuccelli, H.-L. Choi, P. Cho, J.P. How, Real-time multi-UAV task assignment in dynamic and uncertain environments, in: AIAA Guidance, Navigation, and Control Conference, Illinois, Chicago, 2009, pp. 10–13.
[25] N. Geng, Z. Chen, Q.A. Nguyen, D. Gong, Particle swarm optimization algorithm for the optimization of rescue task allocation with uncertain time constraints, Complex Intell. Syst. 7 (2021) 873–890.
[26] N. Sullivan, S. Grainger, B. Cazzolato, Sequential single-item auction improvements for heterogeneous multi-robot routing, Robot. Auton. Syst. 115 (2019) 130–142.
[27] J. Yao, N. Ansari, Online task allocation and flying control in fog-aided internet of drones, IEEE Trans. Veh. Technol. 69 (5) (2020) 5562–5569.
[28] X. Zhou, F. Gao, X. Fang, Z. Lan, Improved bat algorithm for UAV path planning in three-dimensional space, IEEE Access 9 (2021) 20100–20116.

Keqin Li is a SUNY Distinguished Professor of Computer Science with the State University of New York. He is also a National Distinguished Professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has authored or coauthored over 900 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He holds nearly 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top 5 most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an associate editor of the ACM Computing Surveys and the CCF Transactions on High Performance Computing. He has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an AAAS Fellow, an IEEE Fellow, and an AAIA Fellow. He is also a Member of Academia Europaea (Academician of the Academy of Europe).