2019 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, June 11-14, 2019, pp.61-68
DOI: 10.1109/ICUAS.2019.8798293

Efficient Decentralized Task Allocation for UAV Swarms in Multi-target Surveillance Missions

Teng Li, Hyo-Sang Shin and Antonios Tsourdos

***Abstract*— This paper deals with the large-scale task allocation problem for Unmanned Aerial Vehicle (UAV) swarms in surveillance missions. The task allocation problem is proven to be NP-hard which means that finding the optimal solution requires exponential time. This paper presents a practically efficient decentralized task allocation algorithm for UAV swarms based on lazy sample greedy. The proposed algorithm can provide a solution with an expected optimality ratio of at least p for monotone submodular objective functions and of $p(1-p)$ for non-monotone submodular objective functions. The individual computational complexity for each UAV is $O(pr^2)$, where $p \in (0, 0.5]$ is the sampling probability, r is the number of tasks. The performance of the proposed algorithm is testified through digital simulations of a multi-target surveillance mission. Simulation results indicate that the proposed algorithm achieves a comparable solution quality to state-of-the-art algorithms with dramatically less running time. Moreover, a trade-off between the solution quality and the running time is obtained by adjusting the sampling probability.**

***Index Terms*— UAV swarms, task allocation, multi-target surveillance, submodular maximization, lazy sample greedy.**

I. Introduction

Unmanned aerial vehicles (UAVs) possess obvious advantages over manned aircrafts, such as stronger adaptability, smaller size, lower cost, better maneuverability and no pilot casualties etc. In the last few decades, UAVs have been playing important roles in scientific research [1], [2] and industry applications [3], [4]. Tasks that are dull, dirty and dangerous for humans have been completed effectively by UAVs [5], [6]. However, in some missions like large area searching or surveillance might be too challenging to be carried out by only one single UAV. To tackle this problem, a cooperative UAV fleet is considerably a good solution [7], [8]. Nevertheless, new challenges arise among which task allocation is a foundation for mission success. UAVs should be able to figure out an effective and efficient task execution scheme promptly.

This paper focuses on the large-scale decentralized task allocation problem for UAV swarms to carry out the multi-target surveillance mission which is modeled in Section II. The task allocation problem can be considered as maximizing an overall objective function that is the sum of all individual objective function values meanwhile satisfying the constraint that one task can only be allocated to one UAV but each UAV can carry out multiple tasks. In this paper, the objective function of each UAV is formulized as monotone and submodular [9]. The described constraint is also known as the matroid constraint [10], i.e. independence constraint. In this case, task allocation can be defined as submodular welfare problem [11], [12]. The issue with the task allocation problem is that it has been proven to be NP-hard [13] which means that finding the optimal solution requires exponential time. Therefore, it is impractical to calculate the optimal task allocation solution when there are large numbers of tasks and UAVs. Instead, by sacrificing a certain degree of optimality, developing task allocation algorithms that can provide acceptable solution quality with low computational complexity is the main research direction.

Different approaches have been made in order to handle the NP-hardness of the task allocation problem, such as the heuristic approach [14], [15] and the approximation approach [16], [17]. Heuristic algorithms can provide feasible solutions with certain convergence speed. However, this kind of algorithms cannot provide any approximation guarantee of the solution quality. Approximation algorithms are able to achieve mathematical guarantees in terms of both solution quality and computational complexity when the objective functions meet the condition of submodularity [9]. This paper mainly considers approximation algorithms.

The Consensus-Based Bundle Algorithm (CBBA) developed in [18] is one of the most widely accepted state-of-the-art efficient decentralized task allocation algorithms. CBBA uses the concept of Max-Consensus [19], [20] to complete the auction of tasks through the network. In [18], a condition of Diminishing Marginal Gains (DMG) for UAVs' objection functions is formulated. DMG is actually related to the diminishing return which is a basic property of submodularity [9]. With the condition of DMG, CBBA achieves a value approximation guarantee of at least 50% of the optimal solution for maximizing monotone objective functions. However, CBBA fails to provide any approximation guarantee for non-monotone cases.

The Decentralized Sample-based Task Allocation (DSTA) algorithm presented in [21] achieves significant improvements to CBBA [18]. At the beginning of DSTA, each UAV selects tasks from the ground task set with a probability of $p \in (0, 0.5]$ to form its own task samples. In each iteration, each UAV greedily selects the currently best task for itself from its own task samples. Then, all UAVs agree on a globally best task-UAV pair via a maximum consensus protocol and remove this selected task from their task samples in order to prevent conflicts. As a result, DSTA achieves three aspects of improvements compared with CBBA. First, DSTA has obtained expected approximation guarantees for both monotone and non-monotone submodular objective functions. Second,

Teng Li, Hyo-Sang Shin and Antonios Tsourdos are with the School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield MK43 0AL, UK (email: {tengli, h.shin, a.tsourdos}@cranfield.ac.uk)

DSTA reduces the computational complexity because each UAV only considers its own task samples instead of all remaining tasks during each iteration. Third, DSTA is able to adjust the sampling probability in order to balance the trade-off between the approximation ratio and the computational complexity.

The proposed decentralized Lazy Sample based Task Allocation (LSTA) algorithm in this work is a further improvement to DSTA [21] in terms of the practical efficiency meanwhile maintains the aforementioned three advantages compared with CBBA [18]. In theory, LSTA achieves an expected approximation factor of p for monotone submodular objective functions and of $p(1-p)$ for non-monotone submodular objective functions with an individual computational complexity of $O(pr^2)$ for each UAV, where $p \in (0, 0.5]$ is the sampling probability, r is the number of tasks. In practice, the combination of the lazy strategy [22] and the Sample Greedy [9] makes LSTA orders of more efficient than DSTA although they have exactly the same analytical performances.

The performances of the proposed decentralized task allocation algorithm LSTA are verified through digital simulations of a multi-target surveillance mission carried out by a swarm of UAVs. Simulation results indicate that both LSTA and DSTA can provide task allocation solutions that achieve comparably good overall objective function values with orders of less running time compared with those of CBBA. Moreover, LSTA uses significantly less running time than DSTA while achieving nearly the same function values. In addition, a trade-off between the solution quality and running time of the proposed task allocation algorithm is obtained by adjusting the sampling probability.

The rest part of this paper is organized as follows: Section II provides the surveillance mission modeling and necessary preliminaries. Section III presents the proposed task allocation algorithm in details. Section IV demonstrates the performances of LSTA through simulations with a multi-target surveillance mission scenario. Section V summarizes the performances of the proposed task allocation algorithm and the contribution of this work.

II. Problem Statement

This section provides the modeling of the multi-target surveillance mission and presents necessary definitions and basic concepts related to the proposed task allocation algorithm.

A. Mission Modeling

Suppose that there are a set of heterogeneous tasks $j \in \mathcal{T}$ that are randomly located on a $L \times L$ 2-D space. A set of heterogeneous UAVs $a \in \mathcal{A}$ equipped with different sensors are sent to carry out these tasks automatically. The features of the surveillance mission are described in the following.

1) Importance factor: Different tasks have different values that are marked with an importance factor v_j. UAVs tend to give priority to carrying out the tasks that are more valuable than others.

2) Fitness factor: Tasks with different properties require different sensors to be detected effectively. Therefore, the fitness factor m_{aj} reflects the match fitness between the task j and UAV a.

3) Distance discount factor: The mission should be completed as soon as possible. Therefore, with the help of the distance discount factor λ_d, UAVs intend to execute firstly the nearest tasks to them and the lengths of the paths of all UAVs should be balanced.

4) Task amount discount factor: It is usually risky to allocate a large amount of tasks to one UAV but only a few to others. The task amount discount factor λ_n helps to balance the task amounts among all UAVs.

The objective function of each UAV for the surveillance mission can be obtained by combining all the factors that have already been described:

$$f_a(\mathcal{T}_a) = \sum_{j=1}^{|\mathcal{T}_a|} m_{aj} v_j \lambda_d^{\tau(\mathbf{p}_a^j)} \lambda_n^{\sigma(\mathbf{p}_a^j)} \tag{1}$$

where $\mathcal{T}_a \subseteq \mathcal{T}$ is the task list containing all the tasks that are selected by UAV a in sequence, $\mathbf{p}_a$ is the path of UAV a generated according to the order that the tasks appear in $\mathcal{T}_a$, $\mathbf{p}_a^j$ represents the part of $\mathbf{p}_a$ from the initial position of UAV a to the position of task j, $\tau(\mathbf{p}_a^j)$ is the length of the path fragment $\mathbf{p}_a^j$ and $\sigma(\mathbf{p}_a^j)$ is the amount of tasks within the path fragment $\mathbf{p}_a^j$.

Regarding the surveillance mission in this paper, the task allocation problem can be defined as:

Definition 1: (Task Allocation). The task allocation problem is to allocate a set of tasks $\mathcal{T}$ to a set of UAVs $\mathcal{A}$ so as to maximize the total value measured as

$$f(\mathcal{T}, \mathcal{A}) = \sum_{a=1}^{|\mathcal{A}|} f_a(\mathcal{T}_a) \tag{2}$$

meanwhile satisfying the constraint that one task can only be allocated to one UAV but each UAV can carry out multiple tasks.

The overall objective function of the task allocation problem for the surveillance mission can be obtained by combining Eqns. (1) and (2):

$$f(\mathcal{T}, \mathcal{A}) = \sum_{a=1}^{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{T}_a|} m_{aj} v_j \lambda_d^{\tau(\mathbf{p}_a^j)} \lambda_n^{\sigma(\mathbf{p}_a^j)}. \tag{3}$$

B. Preliminaries

Definition 2: (Submodularity [9]) Let $\mathcal{N}$ be a finite set. A real-valued set function $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ is *submodular* if, for all $X, Y \subseteq \mathcal{N}$,

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y).$$

Equivalently, for all $A \subseteq B \subseteq \mathcal{N}$ and $u \in \mathcal{N} \backslash B$,

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B). \tag{4}$$

Eqn. (4) is also known as the diminishing returns which is an important property of submodular functions. Specifically, the marginal gain of a given element u will never increase as more elements have already been selected.

Definition 3: (Marginal gain [9]) For a set function $f : 2^{\mathcal{N}} \to \mathbb{R}$, $S \subseteq \mathcal{N}$ and $u \in \mathcal{N}$, define the *marginal gain* of f at S with respect to u as

$$\Delta f(u|S) := f(S \cup \{u\}) - f(S).$$

Definition 4: (Monotonicity [9]) A set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is *monotone* if, for every $A \subseteq B \subseteq \mathcal{N}$, $f(A) \leq f(B)$. And f is *non-monotone* if it is not monotone.

In this paper, the objective function of the task allocation problem is normalized (i.e. $f(\emptyset) = 0$) and non-negative (i.e. $f(S) \geq 0$ for all $S \subseteq \mathcal{N}$). The objective function Eqn. (3) can be proven to be monotone and submodular.

Definition 5: (Matroid [16]) A matroid is a pair $\mathcal{M} = (\mathcal{N}, \mathcal{I})$ where $\mathcal{N}$ is a finite set and $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is a collection of independent sets, satisfying:

- $\emptyset \in \mathcal{I}$
- $A \subseteq B, B \in \mathcal{I} \Rightarrow A \in \mathcal{I}$
- $A, B \in \mathcal{I}, |A| < |B| \Rightarrow \exists\, b \in B \backslash A$ such that $A \cup \{b\} \in \mathcal{I}$.

Specifically, the matroid constraint includes uniform matroid constraint and partition matroid constraint. The uniform matroid constraint is also called cardinality constraint which is a special case of matroid constraint where any subset $S \subseteq \mathcal{N}$ satisfying $|S| \leq k$ is independent. Partition matroid constraint means that a subset S can contain at most a certain number of elements from each partition.

According to the constraint described in Definition 1, only one task-UAV pair from the task-UAV pairs that are related to this specific task is allowed to be selected. If all task-UAV pairs are considered as a ground set (i.e. $\mathcal{N} := \mathcal{T} \times \mathcal{A}$) and each task-UAV pair as an element of the ground set (i.e. $u_{j,a} := j \times a\ \forall j \in \mathcal{T}, a \in \mathcal{A}$), then the task allocation problem modeled in this paper can be handled as submodular maximization subject to a partition matroid constraint.

III. Algorithm and Analysis

This section describes the proposed task allocation algorithm LSTA in details. In order to better analyze this algorithm, a centralized version of LSTA is presented in Algorithm 3. Since LSTA is an improved version of DSTA, DSTA is introduced from [21] as Algorithm 1 to make LSTA easier to be understood.

A. Algorithms

Let us introduce some notations that are used in the following algorithms. $\mathcal{N}_a$ is the task sample set for UAV a. During the allocation process, each UAV only considers tasks from its own task sample set. $\mathcal{T}_a$ contains the tasks that have already been allocated to UAV a. At the end of these algorithms, $\mathcal{T}_a$ is returned as the task allocation result. In each iteration, j_a^* is the task that can provide the largest marginal value for UAV a, ω_a^* is the largest marginal value for UAV a. In Algorithm 2 and Algorithm 3, $\mathcal{W}_a$ contains the marginal values of all remaining tasks in $\mathcal{N}_a$, ω_a^1 is the first element of the sorted $\mathcal{W}_a$, j_a^1 is the first element of the sorted $\mathcal{N}_a$, $\omega_a(j_{a^*}^*)$ represents the element in $\mathcal{W}_a$ corresponding with the task $j_{a^*}^*$.

The function $MaxCons(a, j_a^*, \omega_a^*)$ in Algorithm 1 and Algorithm 2 is the maximum consensus function [23], which operates the negotiation among all UAVs. UAV a sends its currently best marginal value ω_a^* together with the corresponding UAV id a and task id j_a^* to all other UAVs. At the same time, this UAV also receives the same kind of information from all other UAVs. After gathering all necessary information, it finds the task-UAV pair that provides the globally largest marginal value in the current iteration and returns the corresponding UAV id a^* and task id $j_{a^*}^*$. This process is fully stated in the centralized version of LSTA i.e. Algorithm 3.

Algorithm 1 DSTA for UAV a

```
Input: $f_a : 2^{\mathcal{T}} \to \mathbb{R}_{\geq 0}, \mathcal{T}, p$
Output: A set $\mathcal{T}_a \subseteq \mathcal{T}$
1:  $\mathcal{N}_a \leftarrow \emptyset$, $\mathcal{T}_a \leftarrow \emptyset$
2:  for $j \in \mathcal{T}$ do
3:      with probability $p$,
4:      $\mathcal{N}_a \leftarrow \mathcal{N}_a \cup \{j\}$
5:  end for
6:  while $\exists\ j \in \mathcal{N}_a$ such that $\Delta f_a(j|\mathcal{T}_a) > 0$ do
7:      $j_a^* \leftarrow \arg\max_{j \in \mathcal{N}_a} f_a(j|\mathcal{T}_a)$
8:      $\omega_a^* \leftarrow \Delta f_a(j_a^*|\mathcal{T}_a)$
9:      $a^*, j_{a^*}^* \leftarrow MaxCons(a, j_a^*, \omega_a^*)$
10:     if $a^* == a$ then
11:         $\mathcal{T}_a \leftarrow \mathcal{T}_a \cup \{j_a^*\}$
12:         $\mathcal{N}_a \leftarrow \mathcal{N}_a \backslash \{j_a^*\}$
13:     else
14:         if $j_{a^*}^* \in \mathcal{N}_a$ then
15:             $\mathcal{N}_a \leftarrow \mathcal{N}_a \backslash \{j_{a^*}^*\}$
16:         end if
17:     end if
18: end while
19: return $\mathcal{T}_a$
```

In terms of the structure of these three algorithms, each algorithm consists of two phases, i.e. initialization phase and allocation phase. In the initialization phase of Algorithm 1 (lines 1 $\sim$ 5), UAV a randomly selects tasks from the task ground set $\mathcal{T}$ with probability p to form its own task sample set $\mathcal{N}_a$. Then, in the second phase (lines 6 $\sim$ 19), UAV a only considers tasks that are contained in $\mathcal{N}_a$. UAV a finds the task j_a^* that gives it the largest positive marginal gain ω_a^* given the current $\mathcal{T}_a$ and negotiates with other UAVs through the $MaxCons(a, j_a^*, \omega_a^*)$ function. If UAV a outbids this negotiation, which means a has the globally largest marginal gain, then a will add the task j_a^* into to its task selection set $\mathcal{T}_a$ and remove it from its task sample set $\mathcal{N}_a$. Otherwise, UAV a checks whether the task $j_{a^*}^*$ that provides another UAV with the globally largest marginal gain exists in its own $\mathcal{N}_a$. If $j_{a^*}^*$ does exist in $\mathcal{N}_a$, then a should remove this task from $\mathcal{N}_a$ in order to prevent violation of the constraint. The allocation loop will terminate when no more task can provide a positive marginal gain. Note that only one task can be allocated in each iteration because all UAVs need to reach a consensus on the globally best selection by the principle of greedy algorithms. The sampling operation enables DSTA to achieve an expected approximation guarantee for maximizing non-monotone submodular objective functions and accelerates the task allocation to some degree according to the probability p.

This work further accelerates DSTA by adapting the lazy strategy [22]. In the first phase (lines $1 \sim 13$) of Algorithm 2, similar to DSTA, UAV a also forms its own task sample set $\mathcal{N}_a$ by random sampling. Then, it calculates the marginal values of all tasks in $\mathcal{N}_a$ given the empty $\mathcal{T}_a$ and sorts these tasks in descending order according to their marginal values. In the second phase (lines $14 \sim 40$), UAV a finds the best task with LSTA in a more efficient way than it does with DSTA. Instead of recalculating the marginal values of all remaining tasks, UAV a only recalculates the marginal value of the first task from the sorted task list $\mathcal{N}_a$. If it satisfies the condition that this new marginal value is still larger than the original second one (line 17), then submodularity guarantees that this is the largest marginal value for UAV a in this iteration. Otherwise, resort the task list and recalculate the marginal value of the first task from the resorted task list until it satisfies this condition. Then UAV a negotiates with other UAVs after it finds the best task for itself. The rest of the second phase deals with the conflict resolution by removing the allocated task and its corresponding marginal value from the sorted task list and sorted marginal value list respectively.

The centralized version of LSTA (Algorithm 3) demonstrates the task allocation flow in a clearer way. After all UAVs have found their best tasks in the current iteration, the maximum consensus function $MaxCons$ returns the UAV id a^* and the corresponding task id $j^*_{a^*}$ of the task-UAV pair who can provide the globally largest marginal value as shown in line 24 of Algorithm 3. Then each UAV checks the UAV id and task id to solve the conflict. Lines $32 \sim 34$ in Algorithm 2 and lines $39 \sim 41$ in Algorithm 3 mean that if the globally best task is outbid by another UAV and this task is the first element of the sorted task list $\mathcal{N}_a$ then UAV a needs to recalculate and find the best task for itself again in the next iteration. Otherwise, UAV a can still use the task j^*_a for the negotiation in the next iteration.

B. Analysis

LSTA has exactly the same theoretical performance with DSTA in terms of both approximation ratio and computational complexity. The performances of LSTA are summarized in Theorem 1.

Theorem 1: The proposed algorithm LSTA achieves an expected approximation guarantee of p for monotone submodular objective functions and of $p(1-p)$ for non-monotone submodular objective functions with an expected total computational complexity of $O(pnr)$ and individual complexity of $O(pr^2)$ for each UAV, where $p \in (0, 0.5]$ is the sampling probability, r is the number of tasks i.e. $r = |\mathcal{T}|$ and n is the number of task-UAV pairs i.e. $n = |\mathcal{T}| \times |\mathcal{A}|$.

It is necessary to explain how lazy greedy works before analyzing the reason why LSTA has exactly the same theoretical performances with DSTA. Suppose that $f : 2^{\mathcal{N}} \to \mathbb{R}$ is a submodular function, S_{i-1} and S_i are the selections after the $(i-1)^{th}$ iteration and i^{th} iteration respectively satisfying $S_{i-1} \subset S_i \subset \mathcal{N}$. $R_{i+1} = \mathcal{N} \backslash S_i$ is the set of remaining elements in the $(i+1)^{th}$ iteration. All remaining elements in R_{i+1} are sorted in descending order according to their marginal values given S_{i-1}. u_1 and u_2 are the first and second elements of R_{i+1} respectively. It implies that

$$\Delta f(u_1|S_{i-1}) \geq \Delta f(u_2|S_{i-1}).$$

In the $(i+1)^{th}$ iteration, the new marginal values of all remaining elements should be recalculated given S_i according to the original greed. But in lazy greedy algorithms, only the first element's marginal value given S_i is recalculated. This is the reason why it is called lazy greedy. According to the property of submodularity (Eqn. (4)),

$$\Delta f(u_2|S_i) \leq \Delta f(u_2|S_{i-1}).$$

If it meets the condition that

$$\Delta f(u_1|S_i) \geq \Delta f(u_2|S_{i-1}), \tag{5}$$

then we get

$$\Delta f(u_1|S_i) \geq \Delta f(u_2|S_i).$$

The inequalities are in the same manner for the rest elements from R_{i+1}. Therefore, we can know that the marginal value of u_1 is the largest one in the $(i+1)^{th}$ iteration without recalculating other elements' marginal values. As a result, lazy greedy finds exactly the same element as the original greedy does yet in a more efficient approach. Therefore, LSTA and DSTA achieve the same approximation guarantee.

In the worst case, LSTA and DSTA have the same theoretical computational complexity. If the aforementioned condition (inequality (5)) is not satisfied, the lazy greedy algorithm needs to resort the elements and recalculate the marginal value of the first element of the resorted list. In the worst case, all remaining elements need reevaluating. However, in most cases of real-world applications, only the first or first few elements need to be reevaluated. This is why LSTA is more efficient than DSTA in practice.

Since LSTA has exactly the same theoretical performances with DSTA, please refer to [21] for the proof of Theorem 1. Note that, if $p \in (0.5, 1]$, LSTA achieves an expected approximation guarantee of $1/2$ for monotone submodular objective functions and of $(1-p)/2$ for non-monotone submodular objective functions. According to the theoretical result, as the sampling probability increases, the computational complexity also increases but the approximation guarantee for the non-monotone case decreases. Therefore, there is no advantage on average to set $p \in (0.5, 1]$. When p is set as 0.5, LSTA obtains the best approximation guarantees for both monotone case (50%) and non-monotone case (25%).

Algorithm 2 Decentralized LSTA for UAV a

Input: $f_a : 2^{\mathcal{T}} \rightarrow \mathbb{R}_{\geq 0}, \mathcal{T}, p$
Output: A set $\mathcal{T}_a \subseteq \mathcal{T}$

```
1:  $\mathcal{N}_a \leftarrow \emptyset, \mathcal{T}_a \leftarrow \emptyset, \mathcal{W}_a \leftarrow \emptyset$
2:  for $j \in \mathcal{T}$ do
3:      with probability $p$,
4:      $\mathcal{N}_a \leftarrow \mathcal{N}_a \cup \{j\}$
5:  end for
6:  for $j \in \mathcal{N}_a$ do
7:      $\omega_{aj} \leftarrow \Delta f_a(j|\mathcal{T}_a)$
8:      $\mathcal{W}_a \leftarrow \mathcal{W}_a \cup \{\omega_{aj}\}$
9:  end for
10: Sort $\mathcal{W}_a$ in descending order
11: Sort $\mathcal{N}_a$ according to sorted $\mathcal{W}_a$
12: $\omega_a^* \leftarrow \omega_a^1$
13: $j_a^* \leftarrow j_a^1$
14: while $\exists\ j \in \mathcal{N}_a$ such that $\Delta f_a(j|\mathcal{T}_a) > 0$ do
15:     while $\omega_a^* == 0$ do
16:         $\omega_a^1 \leftarrow \Delta f_a(j_a^1|\mathcal{T}_a)$
17:         if $\omega_a^1 \geq \omega_a^2$ then
18:             $\omega_a^* \leftarrow \omega_a^1$
19:             $j_a^* \leftarrow j_a^1$
20:         else
21:             Resort $\mathcal{W}_a$ and $\mathcal{N}_a$
22:         end if
23:     end while
24:     $a^*, j_{a^*}^* \leftarrow MaxCons(a, j_a^*, \omega_a^*)$
25:     if $a^* == a$ then
26:         $\mathcal{T}_a \leftarrow \mathcal{T}_a \cup \{j_a^*\}$
27:         $\mathcal{N}_a \leftarrow \mathcal{N}_a \backslash \{j_a^*\}$
28:         $\mathcal{W}_a \leftarrow \mathcal{W}_a \backslash \{\omega_a^*\}$
29:         $\omega_a^* \leftarrow 0$
30:     else
31:         if $j_{a^*}^* \in \mathcal{N}_a$ then
32:             if $j_{a^*}^* == j_a^1$ then
33:                 $\omega_a^* \leftarrow 0$
34:             end if
35:             $\mathcal{N}_a \leftarrow \mathcal{N}_a \backslash \{j_{a^*}^*\}$
36:             $\mathcal{W}_a \leftarrow \mathcal{W}_a \backslash \{\omega_a(j_{a^*}^*)\}$
37:         end if
38:     end if
39: end while
40: return $\mathcal{T}_a$
```

Algorithm 3 Centralized LSTA

Input: $f_a : 2^{\mathcal{T}} \rightarrow \mathbb{R}_{\geq 0}\ \forall a \in \mathcal{A}, \mathcal{T}, p$
Output: Sets $\mathcal{T}_a \subseteq \mathcal{T}\ \forall a \in \mathcal{A}$

```
1:  for $a \in \mathcal{A}$ do
2:      $\mathcal{N}_a \leftarrow \emptyset, \mathcal{T}_a \leftarrow \emptyset, \mathcal{W}_a \leftarrow \emptyset$
3:      for $j \in \mathcal{T}$ do
4:          with probability $p$,
5:          $\mathcal{N}_a \leftarrow \mathcal{N}_a \cup \{j\}$
6:      end for
7:      for $j \in \mathcal{N}_a$ do
8:          $\omega_{aj} \leftarrow \Delta f_a(j|\mathcal{T}_a)$
9:          $\mathcal{W}_a \leftarrow \mathcal{W}_a \cup \{\omega_{aj}\}$
10:     end for
11:     Sort $\mathcal{W}_a$ in decreasing order
12:     Sort $\mathcal{N}_a$ according to sorted $\mathcal{W}_a$
13:     $\omega_a^* \leftarrow \omega_a^1$
14:     $j_a^* \leftarrow j_a^1$
15: end for
16: while $\exists\ a \in \mathcal{A}$ and $j \in \mathcal{N}_a$ such that $\Delta f_a(j|\mathcal{T}_a) > 0$ do
17:     for $a \in \mathcal{A}$ do
18:         if $\exists\ j \in \mathcal{N}_a$ such that $\Delta f_a(j|\mathcal{T}_a) > 0$ then
19:             while $\omega_a^* == 0$ do
20:                 $\omega_a^1 \leftarrow \Delta f_a(j_a^1|\mathcal{T}_a)$
21:                 if $\omega_a^1 \geq \omega_a^2$ then
22:                     $\omega_a^* \leftarrow \omega_a^1$
23:                     $j_a^* \leftarrow j_a^1$
24:                 else
25:                     Resort $\mathcal{W}_a$ and $\mathcal{N}_a$
26:                 end if
27:             end while
28:         end if
29:     end for
30:     $a^*, j_{a^*}^* \leftarrow \underset{a \in \mathcal{A}, j_a^* \in \mathcal{N}_a}{\arg\max}\ \omega_a^*(a, j_a^*)$
31:     for $a \in \mathcal{A}$ do
32:         if $a^* == a$ then
33:             $\mathcal{T}_a \leftarrow \mathcal{T}_a \cup \{j_a^*\}$
34:             $\mathcal{N}_a \leftarrow \mathcal{N}_a \backslash \{j_a^*\}$
35:             $\mathcal{W}_a \leftarrow \mathcal{W}_a \backslash \{\omega_a^*\}$
36:             $\omega_a^* \leftarrow 0$
37:         else
38:             if $j_{a^*}^* \in \mathcal{N}_a$ then
39:                 if $j_{a^*}^* == j_a^1$ then
40:                     $\omega_a^* \leftarrow 0$
41:                 end if
42:                 $\mathcal{N}_a \leftarrow \mathcal{N}_a \backslash \{j_{a^*}^*\}$
43:                 $\mathcal{W}_a \leftarrow \mathcal{W}_a \backslash \{\omega_a(j_{a^*}^*)\}$
44:             end if
45:         end if
46:     end for
47: end while
48: return $\mathcal{T}_a\ \forall a \in \mathcal{A}$
```

IV. Simulation Results

This section testifies the proposed task allocation algorithm LSTA for a swarm of UAVs carrying out a multi-target surveillance mission and compares its performance with those benchmark algorithms (CBBA and DSTA) through numerical simulations. The aim of the task allocation in this work is to get as large overall values as possible measured by the objective function (3). Simulations are divided into two phases. The first phase is a demonstration of a task allocation scenario using LSTA with small numbers of UAVs and tasks. The second phase is the performance comparison of LSTA and benchmark algorithms using Monte Carlo simulations with large numbers of UAVs and tasks. In order to simplify the task allocation problem, suppose that physical obstacles are out of consideration, UAVs can automatically avoid collision between each other and the UAV swarm network is strongly connected. Simulation codes are written in Python 3 and simulations are run in a desktop with an Intel i7-6700 3.4GHz CPU and 16GB RAM operated with Win 7.

A. *Task Allocation Demonstration*

Suppose that there are 15 heterogeneous tasks that are randomly located on a $L \times L$ 2-D space ($L = 10km$). The features of these tasks are listed in TABLE I. A fleet of 5 heterogeneous UAVs equipped with different kinds of sensors is sent to carry out the surveillance mission towards these tasks. Different sensors are suitable for different tasks. Hence, each task has different fitness factors with different UAVs. The fitness factors of all task-UAV pairs are listed in TABLE II. The locations of all tasks and UAVs are depicted in Fig. 1. Set the distance discount factor $\lambda_d = 0.95$ and the task amount discount factor $\lambda_n = 0.98$ respectively.

The task allocation results are demonstrated in TABLE III and Fig. 2. The results indicate that the length of paths and the number of tasks for each UAV are distributed in a balanced way. In Fig. 2, UAV5 carries out Task 10 first instead of Task 11 although Task 11 is closer. This is because Task 10 is significantly more important than Task 11 but the difference of the distances is not significant. Meanwhile, the task-UAV fitness factors of task 10 and task 11 with UAV5 are the same. The marginal value of Task 10 is larger than that of Task 11 for UAV5. Similarly, Task 6 is allocated to UAV2 instead of UAV1 because the fitness factor between UAV2 and Task 6 is larger than that between UAV1 and Task 6 which means that UAV2 is more suitable for Task 6. Note that, the task allocation results are varying every time we run the algorithm LSTA due to the random sampling. When there are only a small number of UAVs, it is possible that a task is not selected by any UAV. One solution to this issue is to increase the sampling probability since the efficiency is not the main concern in the low-scale task allocation problems. In the large-scale task allocation problems, if the objective function is monotone, it is unlikely that a task is never selected by any UAV.

TABLE I: Task Features

Task id	Location		Importance Factor
	X/km	Y/km	
1	6.430	2.427	0.8
2	12.990	9.456	0.6
3	5.040	5.118	0.9
4	9.793	9.168	0.6
5	10.350	6.427	1.0
6	4.313	0.797	0.7
7	12.638	1.643	0.7
8	8.369	4.191	1.0
9	7.869	6.883	1.0
10	6.647	8.277	1.0
11	5.110	9.934	0.7
12	11.034	8.161	0.8
13	10.619	3.687	1.0
14	11.385	0.661	0.7
15	13.672	4.785	0.6

TABLE II: Task-UAV Fitness Factors

Task id	UAV1	UAV2	UAV3	UAV4	UAV5
1	1.0	0.9	0.6	0.8	0.9
2	0.9	1.0	0.7	0.9	0.8
3	1.0	0.6	1.0	0.9	0.5
4	0.9	0.8	0.9	1.0	1.0
5	0.6	0.8	0.9	1.0	0.8
6	0.8	1.0	0.7	0.6	0.9
7	0.9	0.6	0.8	1.0	0.7
8	1.0	0.9	0.6	0.8	0.9
9	0.8	0.6	0.9	1.0	0.8
10	0.9	1.0	0.7	0.6	1.0
11	1.0	0.9	0.6	0.1	1.0
12	0.8	0.9	0.7	1.0	0.9
13	0.9	0.8	0.5	0.8	0.9
14	0.5	1.0	0.8	0.9	1.0
15	0.6	0.7	0.9	0.8	0.5

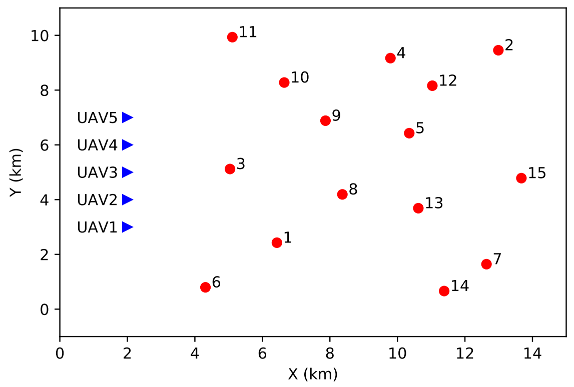


Fig. 1: Locations of Tasks and UAVs

TABLE III: Task Allocation Results

UAV id	Task List	Path Length (km)	Value
1	$8 \to 13 \to 7$	11.658	1.580
2	$6 \to 1 \to 14$	11.883	1.411
3	$3 \to 5 \to 15$	12.217	1.585
4	$9 \to 12 \to 2$	11.694	1.477
5	$10 \to 11 \to 4$	11.825	1.541

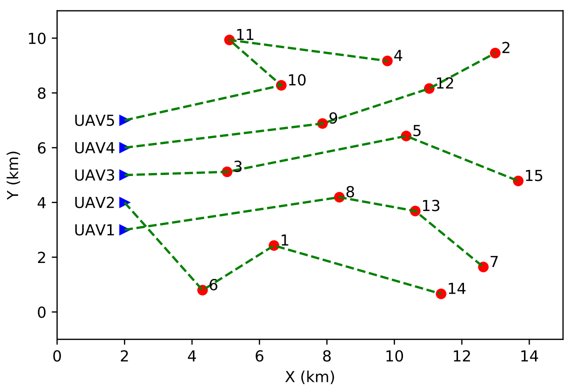


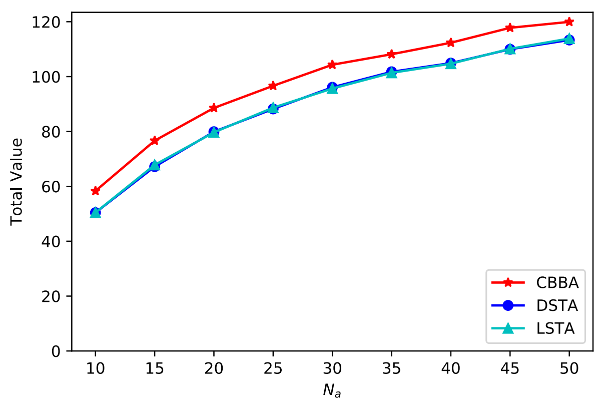
Fig. 2: Task execution scheme

B. Monte Carlo Simulations

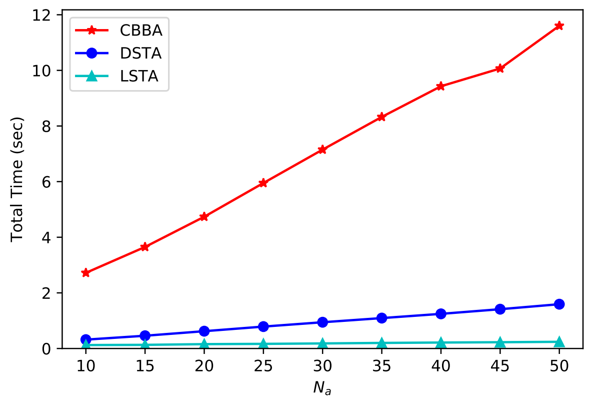
In order to compare the average performance of LSTA and benchmark task allocation algorithms in large-scale task allocation problems, we run 10 rounds of Monte Carlo simulations where tasks and UAVs are randomly placed on a $L \times L$ 2-D space ($L = 10km$). In the simulations, there are 200 tasks and the number of UAVs increases from 10 to 50 which is denoted by N_a. Set the importance factor of each task as a uniformly random number $v_j \in [0.6, 1.0]$ and the match fitness factor of each task-UAV pair as a uniformly random number $m_{aj} \in [0.5, 1.0]$. Set the distance discount factor $\lambda_d = 0.95$ and the task amount discount factor $\lambda_n = 0.98$ respectively. The sampling probability p is set as 0.5 for both DSTA and LSTA.

The simulation results are depicted in Fig. 3. The running time of task allocation for each UAV can be roughly calculated by dividing the total running time with the number of UAVs. As shown in Fig. 3 (b), the running time of all three algorithms increases linearly as the scale of the task allocation problem goes up. In comparison, on the one hand, DSTA and LSTA achieve comparable function values to those of CBBA from approximately 86.5% with 10 UAVs reaching 94.5% with 50 UAVs. On the other hand, DSTA and LSTA use significantly less running time than CBBA as shown in Fig. 3 (b). It is clear from Fig. 3 (a) and (c) that, LSTA obtains almost the same objective function values as DSTA does but is much more efficient. This is a great advantage in the time-limited task allocation applications. Obviously, the proposed algorithm LSTA possesses better scalability than the existing state-of-the-art algorithms for the large-scale task allocation problem in multi-target surveillance missions.

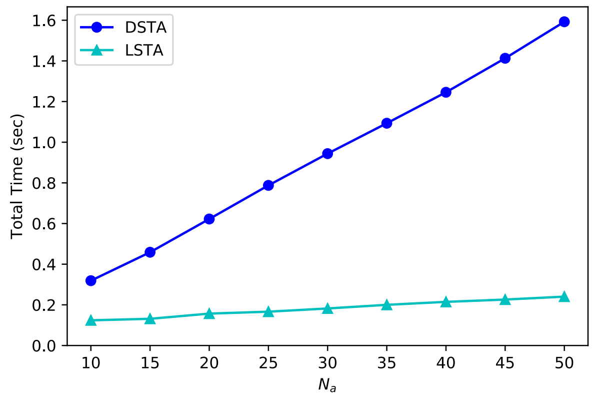
Moreover, we change the sampling probability p of LSTA in order to test the trade-off between the approximation ratio and computational complexity. The selected sampling probabilities are 0.5, 0.3 and 0.1 respectively. Other parameter settings are the same as before.



(a) Total objective function values



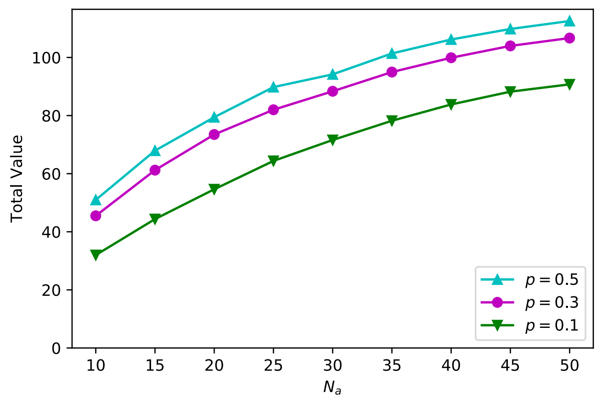
(b) Total running time for CBBA, DSTA and LSTA



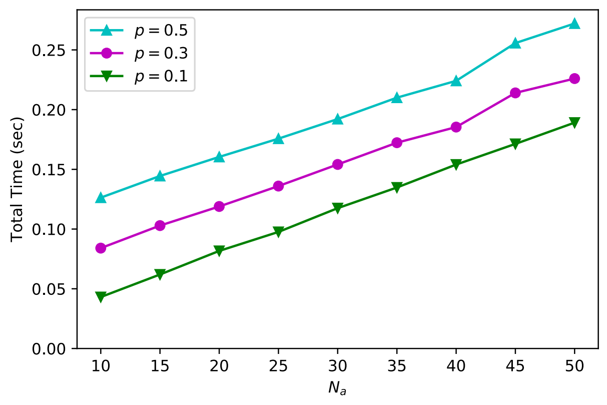
(c) Total running time for DSTA and LSTA

Fig. 3: Performance comparison of different task allocation algorithms

The Monte Carlo simulation results of the trade-off between approximation ratio and computational complexity are demonstrated in Fig. 4. It is clear that as the sampling probability p increases from 0.1 to 0.5, both total objective function values and running time increase. A proper sampling probability can be selected according to the real-world application needs.



(a) Total objective function values



(b) Total running time

Fig. 4: Trade-off for LSTA with varying p

V. Conclusion

This paper presents an efficient decentralized task allocation algorithm for UAV swarms to carry out multi-target surveillance missions. Theoretically, the proposed algorithm LSTA achieves an expected approximation guarantee of p for the surveillance mission modeled in this paper with a computational complexity of $O(pr^2)$ for each UAV. The feasibility of LSTA is testified through numerical Monte Carlo simulations. The simulation results demonstrate that both LSTA and DSTA provide comparably good task allocation solutions but use orders of less running time compared with CBBA. LSTA provides solutions with almost the same quality as DSTA but uses much less running time. Therefore, the proposed algorithm LSTA is more capable than the state-of-the-art algorithms for the large-scale task allocation problems in the multi-target surveillance missions.

Future research would involve the following directions. First, testify the feasibility of LSTA on a reasonable number of decentralized UAVs equipped with normal-performance computational units. Second, Expand the task allocation problem for UAV swarms to the 3D space and consider the uncertainties of surveillance missions.

References

[1] A. C. Watts, V. G. Ambrosia, and E. A. Hinkley, “Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use,” *Remote Sensing*, vol. 4, no. 6, pp. 1671–1692, 2012.

[2] G. Hattenberger, M. Bronz, and M. Gorraz, “Using the paparazzi uav system for scientific research,” in *IMAV 2014, International Micro Air Vehicle Conference and Competition 2014*, 2014, pp. pp–247.

[3] J. Cho, G. Lim, T. Biobaku, S. Kim, and H. Parsaei, “Safety and security management with unmanned aerial vehicle (uav) in oil and gas industry,” *Procedia Manufacturing*, vol. 3, pp. 1343–1349, 2015.

[4] M. Roberts, S. Shah, D. Dey, A. Truong, S. N. Sinha, A. Kapoor, P. Hanrahan, and N. Joshi, “Submodular trajectory optimization for aerial 3d scanning.” in *ICCV*, 2017, pp. 5334–5343.

[5] K. P. Valavanis and G. J. Vachtsevanos, “Future of unmanned aviation,” in *Handbook of unmanned aerial vehicles*. Springer, 2015, pp. 2993–3009.

[6] A. Shukla and H. Karki, “Application of robotics in onshore oil and gas industrya review part i,” *Robotics and Autonomous Systems*, vol. 75, pp. 490–507, 2016.

[7] J. Scherer, S. Yahyanejad, S. Hayat, E. Yanmaz, T. Andre, A. Khan, V. Vukadinovic, C. Bettstetter, H. Hellwagner, and B. Rinner, “An autonomous multi-uav system for search and rescue,” in *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*. ACM, 2015, pp. 33–38.

[8] J. Gu, T. Su, Q. Wang, X. Du, and M. Guizani, “Multiple moving targets surveillance based on a cooperative network for multi-uav,” *IEEE Communications Magazine*, vol. 56, no. 4, pp. 82–89, 2018.

[9] M. Feldman, C. Harshaw, and A. Karbasi, “Greed is good: Near-optimal submodular maximization via greedy optimization,” *arXiv preprint arXiv:1704.01652*, 2017.

[10] R. K. Williams, A. Gasparri, and G. Ulivi, “Decentralized matroid optimization for topology constraints in multi-robot allocation problems,” in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 293–300.

[11] J. Vondrák, “Optimal approximation for the submodular welfare problem in the value oracle model,” in *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 2008, pp. 67–74.

[12] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. Segui, “Decentralised submodular multi-robot task allocation,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 2829–2834.

[13] H.-S. Shin and P. Segui-Gasco, “Uav swarms: Decision-making paradigms,” *Unmanned Aircraft Systems*, p. 397, 2016.

[14] L. Wang, Z. Wang, S. Hu, and L. Liu, “Ant colony optimization for task allocation in multi-agent systems,” *China Communications*, vol. 10, no. 3, pp. 125–132, 2013.

[15] W. Li and W. Zhang, “Method of tasks allocation of multi-uavs based on particles swarm optimization,” *Control and Decision*, vol. 25, no. 9, pp. 1359–1363, 2010.

[16] A. Badanidiyuru and J. Vondrák, “Fast algorithms for maximizing submodular functions,” in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2014, pp. 1497–1514.

[17] N. Buchbinder, M. Feldman, and R. Schwartz, “Comparing apples and oranges: Query trade-off in submodular maximization,” *Mathematics of Operations Research*, vol. 42, no. 2, pp. 308–329, 2016.

[18] H.-L. Choi, L. Brunet, and J. P. How, “Consensus-based decentralized auctions for robust task allocation,” *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[19] R. Olfati-Saber and R. M. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[20] J. Cortés, “Distributed algorithms for reaching consensus on general functions,” *Automatica*, vol. 44, no. 3, pp. 726–737, 2008.

[21] H.-S. Shin, T. Li, and P. Segui-Gasco, “Sample greedy based task allocation for multiple robot systems,” *arXiv:1901.03258*, 2019.

[22] M. Minoux, “Accelerated greedy algorithms for maximizing submodular set functions,” in *Optimization techniques*. Springer, 1978, pp. 234–243.

[23] S. Giannini, A. Petitti, D. Di Paola, and A. Rizzo, “Asynchronous max-consensus protocol with time delays: Convergence results and applications,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 2, pp. 256–264, 2016.