A Deadlock-Free Hybrid Estimation of Distribution Algorithm for Cooperative Multi-UAV Task Assignment With Temporally Coupled Constraints

RUIPENG ZHANG
YANXIANG FENG
YIKANG YANG
Xi’an Jiaotong University, Xi’an, China

XIAOLING LI
Chang’an University, Xi’an, China

This article addresses the cooperative multiunmanned aerial vehicles task assignment problem (CMTAP) with temporally coupled constraints and aims to find a feasible assignment to minimize the equivalent distances of all tasks. We first present a mixed-integer linear programming model of CMTAP. To solve the undesirable deadlocks of CMTAP, a Petri net amender is constructed based on a candidate solution, and a deadlock-free solution is equivalent to a feasible transition sequence that can be fired sequentially in the corresponding amender. With this amender, we present a Petri net-based deadlock amending method (PDAM) with polynomial time complexity to convert a deadlocked solution into a deadlock-free solution. Also, a deadlock-free hybrid estimation of distribution algorithm (DHEDA) is developed for CMTAP by embedding PDAM into the original EDA. To further improve the solution quality, we establish a local exploitation method, and an adaptive operational probability is used to balance the computational burden and local exploitation ability. Then, a match-up-based reassignment method is proposed to cope with time-sensitive targets. Finally, extensive computational experiments demonstrate that PDAM is more effective at solving deadlocks than graph-based methods, particularly for large-scale CMTAP, and DHEDA outperforms existing algorithms when solving CMTAP.

NOMENCLATURE

N_u	Total number of unmanned aerial vehicles (UAVs).
N_t	Total number of targets.
U	Set of UAVs.
D	Set of tasks.
θ_k	Task sequence of k-UAV.
$\boldsymbol{\theta} = \{\theta_k \mid k \in U\}$	Task assignment solution for cooperative multi-UAV task assignment problem.
$\Delta = \{\lambda; \mu\}$	Individual of deadlock-free hybrid estimation of distribution algorithm, where λ implies the execution sequence of tasks and μ represents the assignment information of UAVs.
$(N, M_0) \parallel \Delta$	Petri net amender based on individual Δ.
τ_Δ	Transition sequence extracted from Δ.
Ω_{all}	Set of all individuals.
Ω_e	Set of elite individuals.
N_p	Number of individuals in a population.
$F(\Delta)$	Fitness of individual Δ.
P_{swap}	Probability of swap operations.
P_{insert}	Probability of insert operation.
P_{reorg}	Probability of reorganization operation.
π_s	Detection time of time-sensitive target.
π_d^i	Deadline of the ith time-sensitive target.
$[\pi_{\text{start}}, \pi_{\text{end}}]$	Reassignment horizon.
∂	Replanning time.
$\boldsymbol{\theta}^r$	Reassignment solution.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are widely used in the civilian and military fields, such as search and rescue [1], [2], mineral exploration [3], transportation [4], and military attacks [5], [6], [7]. However, it is difficult for a single UAV to complete complex tasks alone. Thus, multi-UAV systems are being developed to overcome the shortcomings of a single UAV because they can perform tasks cooperatively and in parallel. Task allocation is a critical factor for multi-UAV systems since it determines, which UAVs are assigned to specific targets or tasks. But due to the large solution space or temporal relations among tasks, it is challenging to obtain an effective and feasible task assignment for multi-UAV systems [7].

The cooperative multi-UAV task assignment problem (CMTAP) has recently received increasing attention. Essentially, CMTAP is a nondeterministic polynomial combinatorial optimization problem [9]. Various researchers have proposed a number of mathematical models to solve CMTAP [10], [11], [12], [13], [14]. In particular, Schumacher et al. [10] proposed a mixed-integer linear programming model (MILP) that considers UAV constraints. Evers et al. [11]

Manuscript received 6 May 2022; revised 30 August 2022; accepted 20 November 2022. Date of publication 25 November 2022; date of current version 9 June 2023.

DOI. No. 10.1109/TAES.2022.3224694

Refereeing of this contribution was handled by H. Xu.

This work was supported in part by the Science and Technology Innovation 2030-Key Project of “New Generation Artificial Intelligence” under Grant 2020AAA0108203 and in part by the National Natural Science Foundation of China under Grant 62003258 and Grant 62103062.

Authors’ addresses: Ruipeng Zhang, Yanxiang Feng, and Yikang Yang are with the School of Automation Science and Engineering, Xi’an Jiaotong University, Xi’an 710049, China, E-mail: (zrprepeal@stu.xjtu.edu.cn; fengyanxiang@xjtu.edu.cn; yangyk74@mail.xjtu.edu.cn); Xiaoling Li is with the School of Electronics and Control Engineering, Chang’an University, Xi’an 710064, China, E-mail: (xiaolingli@chd.edu.cn). *(Corresponding author: Yikang Yang.)*

0018-9251 © 2022 IEEE

presented an orienteering problem model and then extended it to fit the dynamics and complexity of UAV missions. Zhao et al. [12] developed a multiagent task assignment model and obtained task assignment results using a Q-learning network. The resulting dynamic programming model is described in [13] and [14].

Existing approaches to solve CMTAP fall into the following two categories: centralized and distributed. The former consists of some optimization algorithms, such as genetic algorithms [15], [16], [17], ant colony algorithms [18], [19], particle swarm optimization algorithms [20], and wolf pack search algorithms [8]. However, these algorithms may easily fall into local optima. For distributed approaches, Choi et al. [23] presented a consensus-based bundle algorithm (CBBA) that uses a market-based decision strategy and uses a consensus routine to produce conflict-free solutions. Wu et al. [24] proposed their method based on the consensus algorithm and the online cooperative strategy, and the Bayesian theorem was used to estimate a target. Kim et al. [25] presented an extension of CBBA and provided a systematic way of grouping the UAVs based on their task preference. These distributed methods achieve a better robustness than centralized methods but struggle with complex constraints.

There is typically complex task precedence when considering multiple tasks. For example, the before relation of tasks refers to that a task must be completed before another begins. If tasks are assigned inappropriately, an undesirable phenomenon called deadlock will occur in the solution. In a deadlock, several UAVs performing tasks fall into a circular-waiting pattern, and all tasks cannot be executed in the worst case [26]. To solve this problem, several studies have investigated deadlock-free task assignments [8], [27], [28], [29], [30], [31]. Specifically, Lemaire et al. [27] dealt with deadlocks by swapping the order of two tasks that violate task precedence constraints. Deng et al. [28] developed a graph theory-based method that detects deadlocks by monitoring the strongly connected components in the task-precedence graph (TPG) and amends them just by matrix transposition. However, too many matrix operations and loops may lead to poor algorithm efficiency. Chen et al. [8] improved the method used in [28] to solve deadlocks by reversing edges instead of using complex matrix operations. Xu et al. [29] proposed a target-bundled genetic algorithm in which deadlocks are avoided using a special encoding method. Customized crossover and mutation operators are used to generate deadlock-free offspring. This approach was enhanced in [30] and [31]. Ye et al. [30] presented a mutation operator with a state-transition scheme to enhance the stochastic searching ability. Tian et al. [31] used a clustering method to sort the targets, through which the computational complexity of the CMTAP was reduced. As a graphical and mathematical modeling tool of discrete-event systems (DESs), Petri nets (PNs) can build state equations, algebraic equations, and other mathematical models of system behavior [33]. These characteristics make PN an efficient mathematical tool that can accurately describe the coupling relation between tasks and UAVs. However, no studies have modeled CMTAP solutions using PNs.

Considering the advantages of PNs in modeling different problems, this study presents a PN amender to resolve the deadlock problem of CMTAP. In this amender, task execution is represented by the firing of corresponding transition, and all tasks are assigned to specific UAVs, but their execution order is not fixed. A deadlock-free solution is equivalent to a feasible transition sequence that contains all transitions and can be fired sequentially in the amender. According to this modeling formalism, we also develop a PN-based deadlock amending method (PDAM) to convert a deadlock individual into a deadlock-free individual. Compared with existing deadlock resolution methods [8], [27], [28], [29], [30], [31], the proposed PDAM provides the following advantages.

1) PDAM has polynomial computational complexity, which means that it can solve deadlocks effectively and quickly, particularly for large-scale CMTAP (see the experiments in Section IV-B).
2) Unlike the graph-based method [8] and [28], PDAM does not need to iterate repeatedly such that no new deadlocks are created during the amending process.
3) PDAM imposes no restrictions on the search mechanism, and the quality of the solutions is not affected, while the methods used in [29], [30], and [31] may fail to do so.

By embedding PDAM into the estimation of distribution algorithm (EDA), we propose an algorithm called the deadlock-free hybrid estimation of distribution algorithm (DHEDA) to solve CMTAP by minimizing the total equivalent distance. EDA is an evolutionary algorithm that reproduces offspring using a probabilistic model instead of individual variation. A potential solution is represented as an individual with two sections: a permutation of tasks and a UAV assignment. To further improve the solution quality, we propose a local exploitation method containing three dedicated operations. Then, an adaptive operational probability is used to balance the computational cost and local exploitation ability. Finally, computational experiments are conducted, and experimental results show that DHEDA outperforms existing algorithms because it provides the best assignment for 32 of 36 instances among the four compared algorithms.

We also extend DHEDA to address the unexpected time-sensitive targets and propose a match-up-based reassignment method, which is confirmed to provide not only good feasible solutions but also advantages in reducing computation time and improving system stability. It also verifies the effectiveness of PDAM on the reassignment problems.

The primary contributions of this study are as follows. First, this article establishes an MILP model for CMTAP with temporally coupled constraints. Second, PDAM is proposed, and its effectiveness and computational complexity in solving deadlocks are demonstrated. Third, based on PDAM, we design DHEDA to solve CMTAP. A dedicated local exploitation method is used to obtain higher quality solutions. Fourth, a match-up-based reassignment method is presented to cope with time-sensitive targets. Last, an experimental scheme is performed to validate the algorithmic capabilities.

The rest of this article is organized as follows. Section II describes the CMTAP model. Section III describes the proposed PDAM method and demonstrates its ability to resolve deadlocks. By embedding PDAM into EDA, we develop the DHEDA. Section IV describes extensive numerical experiments and analyzes their results. Finally, Section V concludes this article.

II. Problem Description and Formulation

This article investigates a CMTAP that consists of a team of UAVs performing different predefined tasks on known targets. Multiple tasks on a target must satisfy temporally coupled constraints. This section presents the details of UAVs, targets, and tasks involved in a CMTAP and yields the mathematical formulation of the considered CMTAP.

A. Unmanned Aerial vehicles

We let $U \equiv \{1, \ldots, N_u\}$ denote a set of N_u UAVs that are initially located at different positions. Each UAV, which is denoted as k-UAV, $k \in U$, has a flying distance limitation ${L_{\mathrm{lim}}}^k$. The number of tasks that k-UAV performs cannot exceed ${N_{\mathrm{lim}}}^k$ due to its capacity limitation, and we let v_k be the speed of k-UAV.

B. Targets and Tasks

We assume that there are N_t ground targets whose positions are known in advance. Each target must be assigned three types of tasks (classification, attack, and verification); thus, there are $3N_t$ tasks in total. For a specific target, the precedence of its three tasks must be satisfied; thus, before a target is attacked by UAVs, classification must be performed, and after the attack is completed, verification is required. To distinguish from the UAVs in U, we denote all tasks by numbers in $D \equiv \{N_u+1, \ldots, 3N_t+N_u\}$.

For a task $j \in D$, let $l = \mathrm{div}(j - N_u - 1, 3) + 1$ and $h = \mathrm{mod}(j - N_u - 1, 3) + 1$, where $\mathrm{div}(a, b)$ takes the quotient when a is divided by b, and the remainder is obtained by $\mathrm{mod}(a, b)$. Then, the j-task belongs to the lth target, and $h = 1, 2, 3$ implies that the j-task is a classification, attack, and verification task, respectively.

We also let $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_{Nu}\}$ denote a task assignment solution, where θ_k, $k \in U$, is a task sequence containing all tasks that must be performed sequentially by k-UAV.

C. Mathematical Model of CMTAP

CMTAP can be considered a multitravel salesman problem (m-TSP) [32], denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \equiv U \cup D$ collects all depot or task nodes and $\mathcal{E}$ contains all edges that represent the shortest traveling path between any two nodes. The k-UAV (or a salesman) is initially located in its depot node $k \in U$, and all nodes in D can be considered to be *cities*. To solve such an m-TSP, we must determine routes for salesmen with the goal of minimizing their total journey cost, and all cities must be visited exactly once. We assume that each UAV leaves a task node immediately after finishing the corresponding task, and it must return to its original depot node after completing the tour.

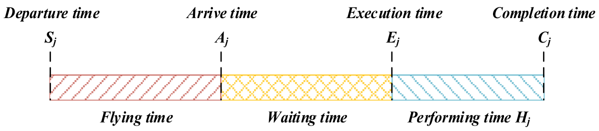


Fig. 1. Timeline of the j-task node.

For a j-task, $j \in D$, some critical time parameters are defined as follows:

1) S_j: the departure time when a UAV flies to the j-task node, which is equal to zero or the time when the UAV leaves the previous node.
2) A_j: the time a UAV arrives at j-task node.
3) E_j: the time when a UAV executes j-task.
4) C_j: the completion time of j-task.
5) H_j: the time required to perform j-task.

The relationship between S_j, A_j, E_j, and C_j is shown in Fig. 1. The interval between S_j and A_j is the flying time, and the waiting time is equal to $E_j - A_j$. Therefore, time interval $E_j - S_j$ is the time consumed before j-task is executed. If the j-task is assigned to k-UAV, we use $(E_j - S_j) \times v_k$ to represent the corresponding equivalent distance.

The mathematical model of the studied CMTAP is presented as follows:

$$\min \sum_{i \in v} \sum_{j \in D} \sum_{k \in U} (E_j - S_j) v_k x_{ij}^k \tag{1}$$

$$\text{s.t.} \sum_{j \in D} x_{kj}^k = 1, k \in U \tag{2}$$

$$\sum_{j \in D} x_{jk}^k = 1, k \in U \tag{3}$$

$$\sum_{k \in U} x_{kj}^k + \sum_{k \in U} \sum_{i \in D} x_{ij}^k = 1, j \in D \tag{4}$$

$$x_{kj}^k + \sum_{i \in D} x_{ij}^k - x_{jk}^k - \sum_{i' \in D} x_{ji'}^k = 0, k \in U, j \in D \tag{5}$$

$$u_j + (3N_t - 2) \sum_{k \in U} x_{kj}^k - \sum_{k \in U} x_{jk}^k \leq 3N_t - 1 \quad \forall j \in D \tag{6}$$

$$\begin{aligned} & u_i - u_j + 3N_t \sum_{k \in U} \\ & x_{ij}^k + (3N_t - 2) \sum_{k \in U} x_{ji}^k \leq 3N_t - 1 \quad \forall i, j \in D, i \neq j \end{aligned} \tag{7}$$

$$\sum_{i \in D} \sum_{j \in v} x_{ij}^k \leq N_{\mathrm{Lim}}^k, k \in U \tag{8}$$

$$\begin{aligned} & \sum_{j \in D} x_{kj}^k L(k, j) + \sum_{i \in D} \\ & \sum_{j \in D} x_{ij}^k L(i, j) + \sum_{j \in D} x_{jk}^k L(k, j) \leq L_{\mathrm{Lim}}^k, k \in U \end{aligned} \tag{9}$$

$$\sum_{k\in U}(S_j - E_i - H_i)x_{ij}^k = 0 \quad \forall i, j \in D \tag{10}$$

$$\sum_{k\in U} S_j x_{kj}^k = 0 \quad \forall j \in D \tag{11}$$

$$A_j - S_j = \frac{\sum_{k\in U}\sum_{i\in v} L(i, j)x_{ij}^k}{v_k} \quad \forall j \in D \tag{12}$$

$$E_{3s+1+N_u} + H_{3s+1+N_u} \le E_{3s+2+N_u} \quad \forall s = \{0, \ldots, N_t - 1\} \tag{13}$$

$$E_{3s+2+N_u} + H_{3s+2+N_u} \le E_{3s+3+N_u} \quad \forall s = \{0, \ldots, N_t - 1\} \tag{14}$$

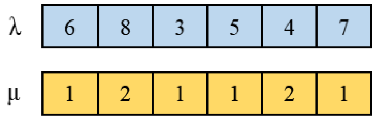
where ${x_{ij}}^k$ is a binary decision variable that describes the task assignment solution $\boldsymbol{\theta}$; ${x_{ij}}^k = 1$ if k-UAV flies from i-node to j-node, and otherwise, ${x_{ij}}^k = 0$; $L(i, j)$ represents the shortest distance between the i-node and j-node, which can be calculated using the Euclidean distance between node coordinates; and u_j is the number of nodes that the UAV has visited until node j. Equation (1) is the objective function, which aims to minimize the total equivalent distances of all tasks; Equations (2) and (3) indicate that each UAV takes off from the corresponding depot node and returns after finishing its tour; Equations (4) and (5) indicate that only one specific UAV arrives and then departs at each task node; Equations (6) and (7) are the traditional subtour elimination constraints (SECs); Equation (8) [resp. (9)] implies the capacity (resp. flying distance) constraint for UAVs; Equation (10) indicates that the departure time of the j-task equals the completion time of the i-task if a UAV is flying from i to j; Equation (11) implies that all UAVs start from their corresponding deports concurrently (i.e., time zero); Equation (12) shows the constraints on flight time for all tasks; and (13) and (14) represent the temporal relationship between classification, attack, and verification tasks on the same target.

III. DEADLOCK-FREE HYBRID ESTIMATION OF DISTRIBUTION ALGORITHM

This article develops a DHEDA to solve CMTAP to minimize the total equivalent distance. The candidate solutions of DHEDA are evaluated by the fitness value, and the probabilistic model is used to extract the features of superior individuals. In each iteration, new individuals are generated to replace some old individuals with the poor performance. A dedicated local exploitation method is used to enhance the solution quality. The primary components of DHEDA are described in the following.

A. Encoder and Initialization

We let $\Delta = \{\lambda;\ \mu\}$ be an individual representing a solution $\boldsymbol{\theta}$ for CMTAP, where $\lambda = (\lambda[1], \lambda[2], \ldots, \lambda[3N_t])$ is a permutation of all tasks in which each task appears exactly once, and $\mu = (\mu[1], \mu[2], \ldots, \mu[3N_t])$ is a sequence of UAVs. For example, we consider an individual $\Delta = \{\lambda;\ \mu\}$ shown in Fig. 2, where $\lambda = (6, 8, 3, 5, 4, 7)$ implies the execution sequence of tasks and $\mu = (1, 2, 1, 1, 2, 1)$ represents the assignment information of UAVs. We assume that the initial population consists of N_p randomly generated individuals.



λ	6	8	3	5	4	7
μ	1	2	1	1	2	1

Fig. 2. Individual for CMTAP with $U = \{1, 2\}$ and $D = \{3, 4, \ldots, 8\}$.

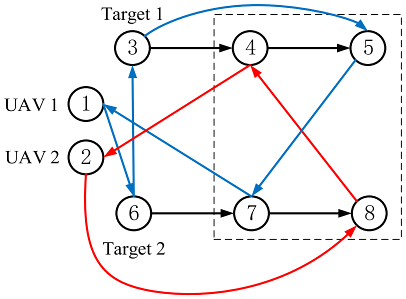


Fig. 3. Digraph representation of a deadlocked solution.

B. Decoder

Given an individual $\Delta = \{\lambda;\ \mu\}$, after applying Algorithm BD, we can obtain a solution $\boldsymbol{\theta}$. However, if tasks are assigned inappropriately, constraints (8) or (9) cannot be satisfied, and deadlocks could arise in solution $\boldsymbol{\theta}$.

```
Algorithm: BD (Basic Decoding Method).
Input: an individual $\Delta = \{\lambda;\ \mu\}$;
Output: a candidate solution $\boldsymbol{\theta} = \{\theta_k \mid k \in U\}$;
Let each $\theta_k = \varnothing$, $k \in U$;
for $m \in \{1, 2, \ldots, 3N_t\}$
    Insert $\lambda[m]$-task to the end of task sequence $\theta_{\mu[m]}$.
end
Output candidate solution $\boldsymbol{\theta}$
```

EXAMPLE 1 Considering $\Delta = \{\lambda;\ \mu\}$ in Fig. 2, the corresponding solution $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$ is obtained by Algorithm BD, where $\theta_1 = <6, 3, 5, 7>$ and $\theta_2 = <8, 4>$. We use a digraph $\mathcal{G} = (\aleph, \wp)$ to represent $\boldsymbol{\theta}$, as shown in Fig. 3, where $\aleph = U \cup D$, the dotted arcs imply the consecutive tasks on the same target, and the task-executing order of UAVs is indicated by solid arcs. Fig. 3 shows that the 8-task is performed before the 4-task, the 7-task is before the 8-task, and the 5-task is before the 7-task; thus, the 5-task must be performed prior to the 4-task, leading to a contradiction with the premise that the 4-task should be completed before performing the 5-task. Thus, a deadlock occurs in $\boldsymbol{\theta}$. Generally, that deadlock can be characterized by a cycle composed of nodes in $\{4, 5, 7, 8\}$.

1) *Constraint Amending Method:* Algorithm CD modifies individual $\Delta = \{\lambda;\ \mu\}$ so that it can meet constraints (8) and (9). We let $\mathrm{dis}(\theta_k)$ be the flight distance of k-UAV to finish all its tasks and let $|\theta_k|$ denote the number of tasks assigned to k-UAV in $\boldsymbol{\theta}$.

Algorithm: CD (Constraint Decoding Method).

```
Input: an individual $\Delta = \{\lambda; \mu\}$;
Output: the modified individual $\Delta' = \{\lambda'; \mu'\}$,
 where (8) and (9) are met.
Obtain solution $\boldsymbol{\theta}$ from $\Delta$ by Algorithm BD;
Let $G = \{u \in U \,|\, |\theta_u| < N_{lim}{}^u\}$ and $X = \{x \in U \,|\, |\theta_x| > N_{lim}{}^x\}$;
while ($X \neq \varnothing$)
  Choose $x \in X$ and a $d$-task $\in \theta_x$, let $p$ be the index of the $d$-task in $\lambda$, i.e., $\lambda[p] = d$;
  Let $u_a = argmin_{u \in G} L(d, u)$, $u_a \in G$, the initial position of $u_a$ is closest to $d$;
  Set $\mu[p] = u_a$;
  Update $\boldsymbol{\theta}$, $G$, and $X$;
end
Let $Y = \{y \in U \,|\, dis(\theta_y) > L_{lim}{}^y\}$;
while ($Y \neq \varnothing$)
  Choose $y \in Y$ and $d'$-task $\in \theta_y$, Let $p'$ be the index of the $d'$-task in $\lambda$, $\lambda[p'] = d'$;
  Let $u_{a'} = argmin_{u \in G} L(d', u)$;
  Set $\mu[p'] = u_{a'}$;
  Update $\boldsymbol{\theta}$, $G$, and $Y$;
end
Let $\lambda' = \lambda$ and $\mu' = \mu$;
Output generated individual $\Delta' = \{\lambda'; \mu'\}$;
```

In Algorithm CD, $X = \{x \in U \mid |\theta_x| > N_{\text{lim}}{}^x\}$ and $Y = \{y \in U \mid \text{dis}(\theta_y) > L_{\text{lim}}{}^y\}$ denote the set of UAVs whose assigned task sequence violates constraints (8) and (9), respectively. If $X \neq \varnothing$, we choose $x \in X$ and $d \in \theta_x$, we let u_a be the UAV in $G = \{u \in U \,|\, |\theta_u| < N_{\text{lim}}{}^u\}$ whose initial position is closest to the d-task, and then, we reassign d to u_a. If $Y \neq \varnothing$, we take a similar action on Y. The output individual Δ' satisfies (8) and (9).

2) *Deadlock-Free Amending Method:* We introduce a PN-based method to resolve deadlocks in CMTAP effectively. PN is a modeling formalism for DESs, and readers are assumed to be familiar with definitions and notations about PNs; for more details, see [33] and the supplemental file [39]. We start by giving a deadlock amender based on an assignment solution of CMTAP.

Definition 1 Given an individual $\Delta = \{\lambda; \mu\}$ for CMTAP, let $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_{Nu}\}$ be the corresponding solution by Algorithm BD. Then, PN deadlock amender $(N, M_0)\|\Delta$ is established by the following steps.

Step 1: For the lth target, we define PN $(\alpha_l, M_l) = (P_l, T_l, F_l, M_l)$, where $P_l = \{p_{ls}, p_{l1}, p_{l2}, p_{le}\}$, $T_l = \{t_{l1}, t_{l2}, t_{l3}\}$, $F_l = \{(p_{ls}, t_{l1}), (t_{l3}, p_{le})\} \cup \{(t_{lh}, p_{lh})\ (p_{lh}, t_{l,h+1}) |\ h \in \{1, 2\}\}$, $M_l(p_{ls}) = 1$, and $M_l(p) = 0$, $\forall p \in P_l \setminus \{p_{ls}\}$.

Step 2: For k-UAV, we define a PN $(\beta_k, M_k) = (\{u_k\}, T_k, F_k, M_k)$, where u_k is the *resource* place corresponding to k-UAV, $T_k = \{t_{lh}\ |, l = \text{div}(j - N_u - 1, 3) + 1, h = \text{mod}(j - N_u - 1, 3) + 1, \forall j \in \theta_k\}$, $F_k = \{(t, u_k), (u_k, t) |\ t \in T_k\}$, $M_k(u_k) = 1$.

Step 3: The amender $(N, M_0)\|\Delta$ is obtained by composing all (α_l, M_l) and (β_k, M_k):
$(N, M_0)\|\Delta = \otimes_{l \in \{1,\ldots,Nt\}} (\alpha_l, M_l) \otimes_{k \in \{1,\ldots,Nu\}} (\beta_k, M_k)$

where operator $\otimes$ represents the composition of two PNs via their common places and transitions.

The subnet (α_l, M_l) established in Step 1 represents the execution procedure of tasks on the lth target. Transitions t_{l1}, t_{l2}, and t_{l3} represent the classification, attack, and verification tasks, respectively, assigned for the lth target. p_{ls} and p_{le} are the source and sink places, respectively, and p_{l1} and p_{l2} represent the intermediate states. Initially, p_{ls} is marked by a unique token, which moves in the ordering $p_{ls} \rightarrow p_{l1} \rightarrow p_{l2} \rightarrow p_{le}$. When the token arrives in p_{le}, all tasks in the lth target are finished.

The subnet (β_k, M_k) established in Step 2 indicates the task assignment of k-UAV. Then, each transition $t_{lh} \in T_k$ in (β_k, M_k), which connects resource place u_k by a pair of directed arcs (t_{lh}, u_k) and (u_k, t_{lh}), denotes that the corresponding task is assigned to k-UAV (i.e., T_k denotes the set of transitions (or tasks) assigned to k-UAV). u_k always contains one token.

Example 2 We reconsider the individual $\Delta = \{\lambda; \mu\}$ in Fig. 2; the corresponding solution is $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$, where $\theta_1 = <6, 3, 5, 7>$ and $\theta_2 = <8, 4>$. By Definition 1, PNs (α_l, M_l), $l \in \{1, 2\}$ are shown in Fig. 4(a). We let u_k be the resource place for k-UAV, $k \in \{1, 2\}$, with a token in it. According to $\boldsymbol{\theta}$, we have $T_1 = \{t_{21}, t_{11}, t_{13}, t_{22}\}$ and $T_2 = \{t_{23}, t_{12}\}$. The corresponding PN (β_k, M_k) for the k-UAV is shown in Fig. 4(b). Finally, the composition of all (α_l, M_l) and (β_k, M_k) is the PN deadlock amender $(N, M_0)\|\Delta$, which is shown in Fig. 4(c).

Next, we present some properties of the PN amender $(N, M_0)\|\Delta$.

Proposition 1 We let $\Delta = \{\lambda; \mu\}$ be an individual of the studied CMTAP. The PN amender $(N, M_0)\|\Delta$ is *safe* and strictly L_1*-live.*

Proof (*1-bounded safe*) There is precisely one token in each place p_{ls} and u_k under the initial marking M_0. Thus, M_0 is 1-bounded safe. We let M be a *safe* reachable marking of $(N, M_0)\|\Delta$ and t be an enabled transition under M. We assume that $M[t > M'$. Due to the safety of M, tokens in each place are no more than one. All arcs in $(N, M_0)\|\Delta$ have a weight of one, and the firing of t adds a token into an empty place or removes a token from a place. Thus, tokens in each place under M' are no more than one, and M' is also 1-bounded safe. Then, using the induction method, all reachable markings of $(N, M_0)\|\Delta$ are found to be safe.

(L_1*-live*) We let u_k and p be the input places of t_{lh}, i.e., ${}^{\bullet}t_{lh} = \{u_k, p\}$. According to Definition 1, u_k is always marked by one token. Thus, if t_{lh} is enabled, there must exist a token in p. The token flow in (α_l, M_l) is unidirectional, and p will

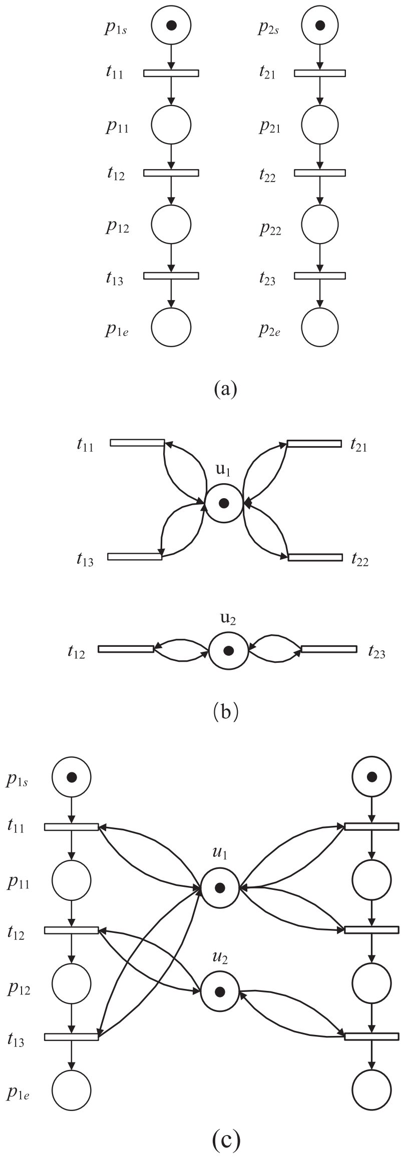


Fig. 4. (a) PNs (α_1, M_1) and (α_2, M_2). (b) PNs (β_1, M_1) and (β_2, M_2). (c) Corresponding PN deadlock amender $(N, M_0)\|\Delta$.

eventually gain a token under a reachable marking, in which t_{lh} can be fired. Conversely, if t_{lh} is fired, p will no longer gain tokens. Therefore, each t_{lh} can be fired exactly once, and $(N, M_0)\|\Delta$ is L_1-live. ♣

According to [33], the safeness and strictly L_1-liveness of $(N, M_0)\|\Delta$ imply that each transition in $(N, M_0)\|\Delta$ can be fired exactly once under initial marking M_0 or under a state M reached from M_0. We let M_E be a state of $(N, M_0)\|\Delta$ satisfying $M_E(p_{le}) = 1$, $\forall l \in \{1, \ldots, N_t\}$, and $M_E(u_k) = 1$, $\forall k \in \{1, \ldots, N_u\}$. To reach M_E from M_0, each transition in $(N, M_0)\|\Delta$ must be fired once all tasks can be performed.

Given an individual $\Delta = \{\lambda;\ \mu\}$, after obtaining the corresponding solution $\boldsymbol{\theta}$ by algorithm BD, we let t_{lh} be the transition in $(N, M_0)\|\Delta$, which corresponds to j-task, as described in Definition 1. Then, a transition sequence $\tau_\Delta = (\tau_\Delta[1], \ldots, \tau_\Delta[3N_t])$ is obtained by replacing each j-task in λ with transition t_{lh}. For example, we consider individual $\Delta = \{\lambda;\ \mu\}$ in Fig. 2, and its corresponding transition sequence $\tau_\Delta = t_{21}t_{23}t_{11}t_{13}t_{12}t_{22}$. If τ_Δ is *feasible* under M_0 (i.e., M_E can be reached from M_0 through τ_Δ, which is denoted as $M_0[\tau_\Delta > M_E)$, then all underlying tasks can be finished through firing transitions in τ_Δ sequentially, and the corresponding solution $\boldsymbol{\theta}$ is deadlock-free. Thus, the deadlock in Δ equivalents is detected to determine whether its corresponding transition sequence τ_Δ is feasible under M_0.

Based on the abovementioned analysis, we develop PDAM to resolve the deadlock problem in an individual $\Delta = \{\lambda;\ \mu\}$ based on the amender $(N, M_0)\|\Delta$.

```
Algorithm: PDAM (PN-Based Deadlock Amending Method).
  Input: an individual $\Delta = \{\lambda;\ \mu\}$;
  Output: a deadlock-free solution $\Delta^*$;
  1:  Generate PN amender $(N, M_0)\|\Delta$ based on $\Delta$;
  2:  Generate the transition sequence $\tau_\Delta$;
  3:  for $n = 1$ to $3N_t$
  4:     while($\tau_\Delta[n]$ is disabled under $M_{n-1}$)
  5:        Remove $\tau_\Delta[n]$ to the end of $\tau_\Delta$;
  6:        Remove $\mu[n]$ to the end of $\mu$;
  7:     end
  8:     Let $M_{n-1}[\tau_\Delta[n] > M_n$;
  9:  end
  10: Let ${\tau_\Delta}^* = \tau_\Delta$ and $\mu^* = \mu$;
  11: Generate a task permutation $\lambda^*$ according to
      ${\tau_\Delta}^*$;
  12: Output $\Delta^* = (\lambda^*;\ \mu^*)$;
```

Given an individual Δ, PDAM first generates its amender $(N, M_0)\|\Delta$ and transition sequence τ_Δ, as shown in Lines 1–2. Then, in the loop of Lines 4–7, we detect whether $\tau_\Delta[n]$ is enabled under marking M_{n-1} or not; if it is enabled, we fire it and generate a new marking M_n (i.e., $M_{n-1}[\tau_\Delta[n] > M_n$). Otherwise, $\tau_\Delta[n]$ (resp. $\mu[n]$) is moved to the end of τ_Δ (resp. μ), and the loop is continued until finding an enabled transition under M_{n-1}. When the iteration in Lines 3–9 ends, a transition sequence ${\tau_\Delta}^*$ is generated such that $M_0[{\tau_\Delta}^* > M_E$, as well as a new permutation μ^*. We can generate a task permutation λ^* from ${\tau_\Delta}^*$. Then, the new individual $\Delta^* = (\lambda^*, \mu^*)$ corresponds to a deadlock-free solution. The following result establishes the computational complexity of PDAM.

PROPOSITION 2 PDAM has polynomial time complexity and can obtain a deadlock-free task allocation solution.

PROOF By Proposition 1, amender $(N, M_0)\|\Delta$ is strictly L_1-live, and each transition t_{ih} can be fired exactly once under some marking M_h before reaching M_E. This fact means that there is at least one enabled transition under M_h. Such an enabled transition is found through the iteration in PDAM. Finally, M_E is reached, and a feasible transition sequence ${\tau_\Delta}'$ is obtained. Thus, the corresponding solution Δ' generated from ${\tau_\Delta}'$ is deadlock-free.

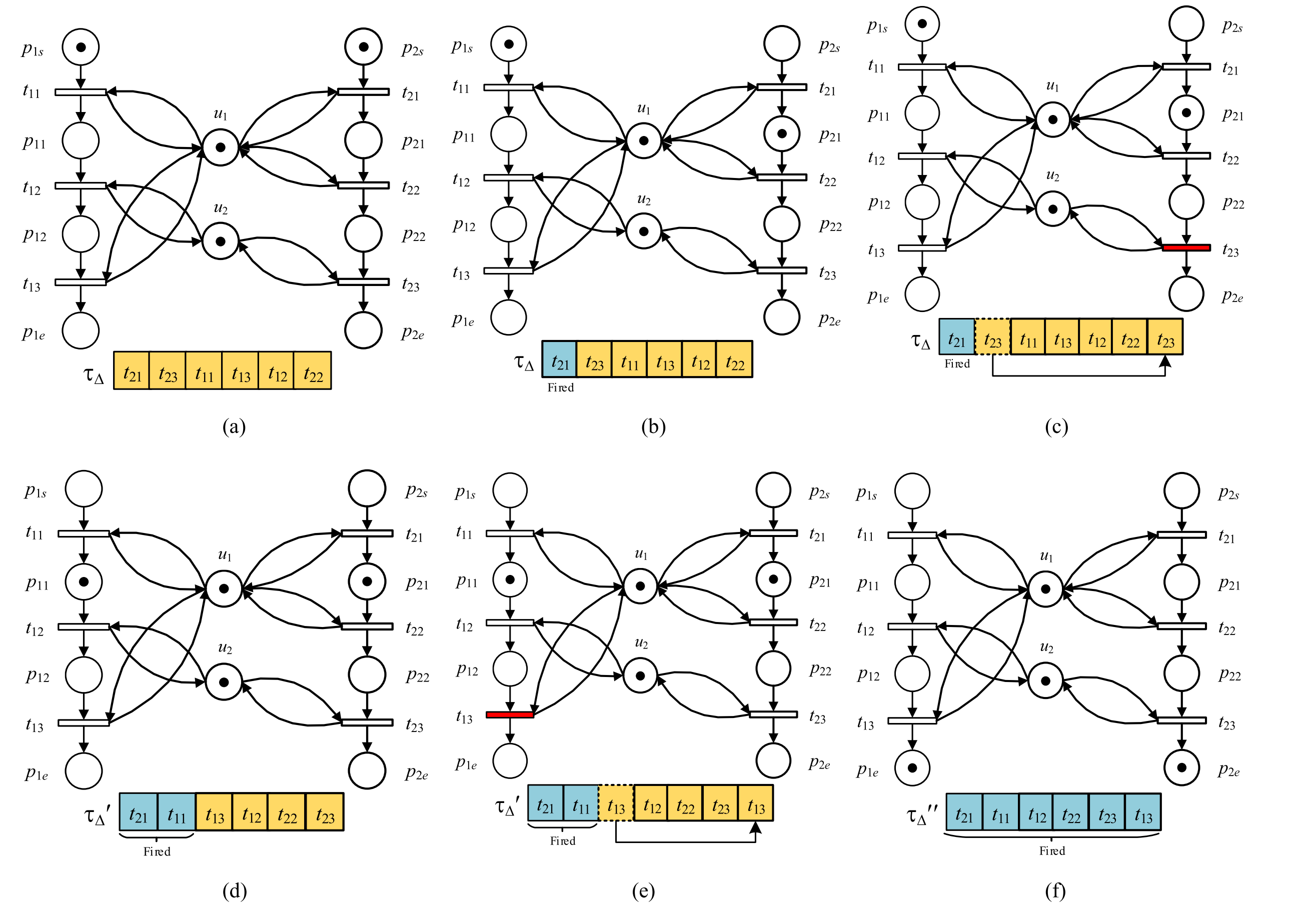

p_{1s}
t_{11}
p_{11}
t_{12}
p_{12}
t_{13}
p_{1e}
p_{2s}
t_{21}
p_{21}
t_{22}
p_{22}
t_{23}
p_{2e}
u_1
u_2
τ_Δ t_{21} t_{23} t_{11} t_{13} t_{12} t_{22}
(a)
τ_Δ t_{21} t_{23} t_{11} t_{13} t_{12} t_{22}
Fired
(b)
τ_Δ t_{21} t_{23} t_{11} t_{13} t_{12} t_{22} t_{23}
Fired
(c)
τ_Δ' t_{21} t_{11} t_{13} t_{12} t_{22} t_{23}
Fired
(d)
τ_Δ' t_{21} t_{11} t_{13} t_{12} t_{22} t_{23} t_{13}
Fired
(e)
τ_Δ'' t_{21} t_{11} t_{12} t_{22} t_{23} t_{13}
Fired
(f)

Fig. 5. Processing of PDAM. (a) Initial marking M_0 in $(N, M_0)\|\Delta$. (b) t_{21} is fired, $M_0[t_{21}> M_1$. (c) t_{23} is disabled under marking M_1. (d) t_{11} is fired, $M_1[t_{11}>M_2$. (e) t_{13} is disabled under marking M_2. (f) Final marking M_E.

The entire method repeats $3N_t$ times. In the nth iteration, at most $(3N_t - n)$ transitions are checked. Thus, the complexity of the PDAM is $O(N_t{}^2)$, i.e., PDAM is polynomial. ♣

Example 3 The process of applying PDAM to the individual $\Delta = \{\lambda; \mu\}$ in Fig. 2 is shown in Fig. 5. The corresponding $\tau_\Delta = t_{21}t_{23}t_{11}t_{13}t_{12}t_{22}$ and amender $(N, M_0)\|\Delta$ are shown in Fig. 5(a). Transition t_{21}, the first transition in τ_Δ, is enabled under M_0. After firing t_{21} in $(N, M_0)\|\Delta$, a new marking M_1 is reached from M_0 (i.e., $M_0[t_{21} >M_1$), as shown in Fig. 5(b). However, the second transition $\tau_\Delta[2] = t_{23}$ is disabled under M_1, as shown in Fig. 5(c). Then, we move t_{23} (resp. 2) to the end of τ_Δ (resp. μ), and a new transition sequence $\tau_\Delta{}' = t_{21}t_{11}t_{13}t_{12}t_{22}t_{23}$ and $\mu' = (1, 1, 1, 2, 1, 2)$ are obtained. For the second transition $\tau_\Delta{}'[2] = t_{11}$, because it is enabled under M_1, we have $M_1[t_{11} >M_2$, as shown in Fig. 5(d). Because $\tau_\Delta{}'[3] = t_{13}$ is disabled under M_2, we take a similar action and obtain $\tau_\Delta{}'' = t_{21}t_{11}t_{12}t_{22}t_{23}t_{13}$ and $\mu'' = (1, 1, 2, 1, 2, 1)$. After sequentially firing transitions t_{12}, t_{22}, t_{23}, and t_{13} from M_2, the marking M_E is reached (i.e., $M_2[t_{12}t_{22}t_{23}t_{13} >M_E$), as shown in Fig. 5(e). Thus, M_E can be reached from M_0 through $\tau_\Delta{}''$, i.e., $M_0[\tau_\Delta{}'' >M_E$. Based on $\tau_\Delta{}''$ and μ'', we finally obtain a new deadlock-free individual $\Delta^* = \{\lambda^*; \mu^*\}$, where $\lambda^* = (6, 3, 4, 7, 8, 5)$ and $\mu^* = (1, 1, 2, 1, 2, 1)$.

3) *Individual Amending Method:* An individual amending method is shown in Algorithm IAD by synthesizing the above two amending algorithms.

Algorithm: IAD (Individual Amending Method).

```
Input: an individual $\Delta$;
Output: a feasible individual $\Delta'$;
while (8) or (9) is not met in $\Delta$ or $\Delta$ is deadlocked)
    Let $\Delta$ be the input of Algorithm CD and $\Delta_1$ be the result;
    Let $\Delta_1$ be the input of PDAM and $\Delta_2$ be the result;
    Let $\Delta = \Delta_2$;
end
Let $\Delta' = \Delta$;
Output feasible individual $\Delta'$;
```

In Algorithm IAD, after running Algorithm CD on Δ, a new Δ_1 meeting (8) and (9) is obtained. Then, the PDAM is executed to obtain the deadlock-free individual Δ_2. The resulting Δ_2 may violate (9), and thus, a loop (Lines 1−5) is used. The output Δ' satisfies all constraints and has no deadlocks.

C. Computation of Objection Function

For a deadlock-free individual $\Delta = \{\lambda;\ \mu\}$, we let $\boldsymbol{\theta} = \{\theta_k, k \in U\}$ be the corresponding solution by Algorithm BD. Then, the parameters associated with any j-task, $j \in \theta_k$, can be calculated by the method described in this section.

First, we let $\chi(j) \in D$ be the preceding task of the j-task according to the task orders in θ_k. If j-task is the first element of θ_k, set $\chi(j) = 0$; otherwise, k-UAV begins flying to the location of j-task immediately upon the completion of task $\chi(j)$. Thus, we have

$$S_j = \begin{cases} 0, & \text{if } \chi(j) = 0 \\ C_{\chi(j)}, & \text{otherwise} \end{cases}. \tag{15}$$

The time that k-UAV arrives at the location of j-task can be calculated as follows:

$$A_j = \begin{cases} S_j + \frac{L(k,j)}{v_k}, & \text{if } \chi(j) = 0 \\ S_j + \frac{L(i,j)}{v_k}, & \text{otherwise} \end{cases}. \tag{16}$$

We let $\gamma(j)$ be the previous task of j according to the temporal relation. Considering CMTAP in Example 1, tasks 3, 4, and 5 represent the classification, attack, and verification tasks for the first target, respectively. We have $\gamma(4) = 3$ and $\gamma(5) = 4$. Tasks $\gamma(j)$ and j are performed on the same target, but the latter begins only after the former is finished. Therefore, we have

$$E_j \geq E_{\gamma(j)} + H_{\gamma(j)} \tag{17}$$

where $H_{\gamma(j)}$ is the performing time of task $\gamma(j)$.

For $\Delta = \{\lambda;\ \mu\}$, let $\omega(j) \in D$ be the preceding task of j-task in λ. If the j-task is the first element of λ, we set $\omega(j) = 0$. Considering $\lambda = (6, 8, 3, 5, 4, 7)$ in Fig. 2, we have $\omega(8) = 6$, $\omega(3) = 8$, and $\omega(6) = 0$. We assume that the execution time of j-task is no less than $\omega(j)$-task, which is consistent with the firing rule of the corresponding transition sequence in a PN amender. Thus, we have

$$E_j \geq E_{\omega(j)}. \tag{18}$$

It is clear that $E_j \geq A_j$. Based on the abovementioned inequalities, we can give the calculation formula of E_j, as shown in (19)

$$E_j = \max\{A_j, E_{\gamma(j)} + H_{\gamma(j)}, E_{\omega(j)}\}. \tag{19}$$

The completion time C_j of j-task is

$$C_j = E_j + H_j. \tag{20}$$

For a deadlock-free individual $\Delta = \{\lambda;\ \mu\}$, we can calculate S_j, A_j, E_j, and C_j for each j-task in λ sequentially using (15), (16), (19), and (20), respectively. We let the fitness of individual Δ be equal to the value of objective function (1), which is the sum of equivalent distances of all tasks, is denoted by $F(\Delta)$, and can be computed by

$$F(\Delta) = \sum_{u=1}^{3N_t} (E_{\lambda[u]} - S_{\lambda[u]}) v_{\mu[u]}. \tag{21}$$

D. Probabilistic Model

The probabilistic model is an essential component of determining the performance of EDA that describes the distribution of superior individuals and captures the features that make them better than others.

We let $\Omega_{\mathbf{all}}$ denote the population in EDA, which consists of N_p individuals. Let Ω_e be the elite set containing the best $[N_p \times \delta]$ individuals, where δ is a proportion parameter and $[N_p \times \delta]$ indicates the maximum integer less than $N_p \times \delta$.

The voting procedure in [35] is used to construct the probability model from each elite individual $\Delta_q = \{\lambda_q;\ \mu_q\} \in \Omega_e$. Specifically, we construct a $3N_t$-dimensional square $\Psi_{\lambda q}$ for λ_q: for $i \in [1, 3N_t]$, set $\Psi_{\lambda q}(k{-}N_u, i) = 1$, where $k = \lambda_q[i]$, and other elements in $\Psi_{\lambda q}$ are set to zero. In this study, $\Psi_{\lambda q}(m, n) = 1$ represents that $(m + N_u)$-task is in the nth position of λ_q. Similarly, another $N_u \times 3N_t$ matrix $\Psi_{\mu q}$ is established: for each $i \in [1, N_u]$, we set $\Psi_{\mu q}(k, i) = 1$, where $k = \mu_q[i]$, and other elements in $\Psi_{\mu q}$ are zero. $\Psi_{\mu q}(m, n) = 1$ represents that m-UAV is in the nth position of μ_q.

Also, the dominance matrix Q_λ (resp. Q_μ) can be calculated from $\Psi_{\lambda q}$ (resp. $\Psi_{\mu q}$) as follows:

$$Q_\lambda(m, n) = \sum_{\Delta q \in \Omega_e} \Psi_{\lambda_q}(m, n) \times \left(\exp\left(\frac{F(\Delta_w) - F(\Delta_q)}{F(\Delta_w) - F(\Delta_b)} \right) - 1 \right) \tag{22}$$

$$Q_\mu(m, n) = \sum_{\Delta q \in \Omega_e} \Psi_{\mu_q}(m, n) \times \left(\exp\left(\frac{F(\Delta_w) - F(\Delta_q)}{F(\Delta_w) - F(\Delta_b)} \right) - 1 \right) \tag{23}$$

where Δ_w (resp. Δ_b) represents the worst (resp. best) individual, and the entry $Q_\lambda(m, n)$ (resp. $Q_\mu(m, n)$) denotes the weighted times that $(m + N_u)$-task (resp. m-UAV) appears at the nth position in λ_p (resp. μ_q) of all elite individuals.

EXAMPLE 6 We consider the CMTAP in Example 1. Given a population $\Omega_{\mathbf{all}} = \{\Delta_1, \ldots, \Delta_5\}$ containing five individuals, their details are as follows:

$$\Delta_1 = \{\lambda_1;\ \mu_1\} = \{(6, 7, 8, 3, 4, 5);\ (2, 1, 2, 1, 2, 1)\},\quad F(\Delta_1) = 187.89;$$

$$\Delta_2 = \{\lambda_2;\ \mu_2\} = \{(6, 7, 3, 8, 4, 5);\ (1, 2, 2, 1, 2, 1)\},\quad F(\Delta_2) = 206.94;$$

$$\Delta_3 = \{\lambda_3;\ \mu_3\} = \{(8, 6, 5, 3, 4, 7);\ (2, 1, 1, 2, 1, 2)\},\quad F(\Delta_3) = 212.77;$$

$$\Delta_4 = \{\lambda_4;\ \mu_4\} = \{(6, 7, 3, 4, 5, 8);\ (2, 1, 2, 2, 1, 1)\},\quad F(\Delta_4) = 216.94;$$

$$\Delta_5 = \{\lambda_5; \mu_5\} = \{(3, 6, 4, 7, 8, 5); (2, 1, 1, 2, 2, 1)\},$$
$$F(\Delta_5) = 228.67.$$

We assume that $\delta = 0.4$; then, the elite set Ω_e contains the 2 best individuals Δ_1 and Δ_2 (i.e., $\Omega_e = \{\Delta_1, \Delta_2\}$). For $\Delta_1 \in \Omega_e$, because $\lambda_1[1] = 6$ and $N_u = 2$, we can obtain $\Psi_{\lambda 1}(4, 1) = 1$. $\Psi_{\lambda 1}(5, 2) = \Psi_{\lambda 1}(6, 3) = \Psi_{\lambda 1}(1, 4) = \Psi_{\lambda 1}(2, 5) = \Psi_{\lambda 1}(3, 6) = 1$, and the other elements in $\Psi_{\lambda 1}$ are zero. Similarly, $\Psi_{\mu 1}$ can also be obtained

$$\Psi_{\lambda_1} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$
$$\Psi_{\mu_1} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

$\Delta_b = \Delta_1$ and $\Delta_w = \Delta_5$, and we set all elements in Q_λ and Q_μ to zero at first. We then consider individual Δ_1. Because $\Psi_{\lambda 1}(4, 1) = 1$, its weight $\exp(\frac{F(\Delta_w)-F(\Delta_1)}{F(\Delta_w)-F(\Delta_b)}) - 1 = 1.72$. Thus, Δ_1s votes for $Q_\lambda(4, 1)$ are 1.72, and $Q_\lambda(4, 1)$ increases accordingly. After all individuals in Ω_e have voted, the final dominance matrices Q_λ and Q_μ are described by follows:

$$Q_\lambda = \begin{bmatrix} 0 & 0 & 0.7 & 1.72 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.42 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.42 \\ 2.44 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.44 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.72 & 0.7 & 0 & 0 \end{bmatrix}$$
$$Q_\mu = \begin{bmatrix} 0.70 & 1.72 & 0 & 2.42 & 0 & 2.42 \\ 1.72 & 0.70 & 2.42 & 0 & 2.42 & 0 \end{bmatrix}.$$

E. Offspring Reproduction and Replacement

This section extracts ζ individuals from the current population to construct a set of seeds Ω_{new} by the roulette method. Each offspring $\Delta = \{\lambda; \mu\}$ is reproduced from each seed $\Delta_n = \{\lambda_n; \mu_n\} \in \Omega_{\text{new}}$ as follows.

First, element $\lambda_n[l]$ is scheduled to the lth position of λ with a probability $p_l = \alpha \times \beta \times \exp\{(F(\Delta_b) - F(\Delta_n)) / (F(\Delta_w) - F(\Delta_b))\}$, where $\alpha = 1/(e-1)$ is a constant, and parameter $\beta = \exp\{(F(\Delta_{b'}) - F(\Delta_b))/F(\Delta_{b'})\}$ is associated with the best individual $\Delta_{b'}$ of the previous generation. Then, we let Λ_n be the set of tasks that are not assigned to λ, thus far. For each unassigned qth position in λ, we randomly select j-task $\in \Lambda_n$ with probability $p_\lambda = Q_\lambda(j - N_u, q)/\Sigma_{p \in \Lambda n} Q_\lambda(p - N_u, q)$, where $Q_\lambda(m, n)$ is the (m, n)-entry of Q_λ. This process is repeated until $\Lambda_n = \varnothing$ and all elements in λ have been assigned a task. Finally, for each rth position in μ, we randomly select k-UAV $\in U$ with probability $P_\mu = Q_\mu(k, r)/\Sigma_{p \in U} Q_\mu(p, r)$ and set $\mu[r] = k$. The obtained offspring Δ is amended with Algorithm IAD, and if this Δ is better than the worst ζ individuals of the current population and different from others, this offspring is reserved. This replacement preserves superior genes and ensures the diversity of the population.

F. Local Exploitation Method

This section proposes a dedicated local exploitation method to further improve the solution quality. We first define r_z as the probability of performing local exploitation at the zth iteration

$$r_z = \begin{cases} 0.01, & \text{when } R_z \leq 0.01 \\ R_z, & \text{when } 0.01 < R_z \leq 1 \end{cases} \tag{24}$$

where $R_z = \exp(\beta/\alpha)$, $\alpha = \frac{F_z - F_1 - 1}{F_1}$, and $\beta = \frac{F_{z-1} - F_z}{F_z}$. In this study, F_1, F_z, and $F_{z\text{-}1}$ represent the fitness values of the best individuals in the initial, current, and previous populations, respectively. If there is a better individual in the zth generation population, $F_z < F_{z-1}$; otherwise, $F_z = F_{z-1}$. Also, we obtain $F_1 \geq \cdots \geq F_{z-1} \geq F_z$, and α is definitely negative. Thus, the smaller β is, the more the algorithm needs to perform local exploitation, and therefore, the larger R_z and probability indicator r_z are. In addition, if R_z is small, we set $r_z = 0.01$ to strengthen the exploitation ability.

The introduction of r_z can bring a reasonable balance between local exploitation ability and computational cost. When the algorithm fails to achieve a better solution, the dedicated local exploitation method is always called. The method is introduced in the following.

We start by selecting ζ individuals from the population Ω_{all} to create a set Ω_a by the roulette method. Three operations (swap, insert, and reorganization) are then performed on each individual $\Delta \in \Omega_a$ with probability P_{swap}, P_{insert}, and P_{reorg}, respectively. The supplemental file [39] details all operations.

Swap operation: For an individual $\Delta = \{\lambda; \mu\} \in \Omega_a$, we choose two different h-task and j-task from λ and then swap them. Because the results may be infeasible, we perform Algorithm IAD to ensure that the obtained individual is deadlock-free and satisfies constraints.

Insert operation: Given an individual $\Delta = \{\lambda; \mu\} \in \Omega_a$, two different h-task and j-task are selected from λ, h-task is inserted into the position of j-task, while the elements after j-task in λ are shifted backward. Algorithm IAD is also performed on the generated individual.

Reorganization operation: Given an individual $\Delta = \{\lambda; \mu\} \in \Omega_a$, we choose another individual $\Delta_y = \{\lambda_y; \mu_y\}$ from Ω_a. Then, individual Δ is reorganized by replacing μ with μ_y (i.e., $\Delta = \{\lambda; \mu_y\}$). We also perform Algorithm IAD on it.

All individuals obtained by performing the above three operations are collected in $\Omega_a{}'$. The worst ζ individuals in Ω_{all} are replaced by the best ζ ones in $\Omega_a \cup \Omega_a{}'$.

G. Overall DHEDA

By embedding PDAM and the local exploitation method into EDA, we obtain DHEDA, as detailed in the following.

Algorithm: DHEDA (Deadlock-Free Hybrid Estimation of Distribution Algorithm).

```
Input: Parameters $N$, $\delta$, $P_{swap}$, $P_{insert}$, $P_{reorg}$, $\zeta$, and the iteration number $T_{max}$.
Output: the optimal solution $\boldsymbol{\theta}$.
Generate initial population $\Omega_{all}$ with $N$ individuals and set $z = 1$;
Perform Algorithm IAD on each individual in $\Omega_{all}$;
  while $z < T_{max}$
  Obtain elite set $\Omega_e \subseteq \Omega_{all}$ based on fitness value with ratio $\delta$;
  Obtain dominance matrices $Q_\lambda$ and $Q_\mu$ according to $\Omega_e$;
  Update the population $\Omega_{all}$ according to $Q_\lambda$ and $Q_\mu$;
  Compute the probability $r_z$ for the current $z$-th iteration;
  if rand(0, 1) $< r_z$ //performing local exploitation
    Obtain antibody set $\Omega_a$ with $\zeta$ individuals from $\Omega_{all}$ by the roulette method;
    Perform Algorithm SO, IO, RO on each individual in $\Omega_a$. Let $\Omega_a{}'$ be the set of all obtained individuals;
    The worst $\zeta$ individuals in $\Omega_{all}$ are replaced by the best $\zeta$ ones in $\Omega_a \cup \Omega_a{}'$;
  end
  $z = z + 1$;
  end
Let $\Delta_{best}$ be the optimal individual in $\Omega_{all}$, and generate a solution $\boldsymbol{\theta}$ from $\Delta_{best}$;
Output solution $\boldsymbol{\theta}$;
```

H. Uncertainty Problem in CMTAP

The time-sensitive targets with time window constraints are unpredictable disturbances in the dynamic environment, and their appearance will make the original assignment infeasible. To solve this problem, we propose a match-up [36] based reassignment method, which consists of the following three steps: setting the reassignment horizon, performing the reassignment, and integrating the result into the original solution.

Assume that at time π_s, N_n new targets are detected, each requiring three tasks described in Section II, and the tasks of the ith target should be completed within $[\pi_s, \pi_d{}^i]$. All these new tasks are denoted as $D_n \equiv \{3N_t + N_u + 1, \ldots, 3N_t + N_u + 3N_n\}$. Let the replanning time ∂ be a smaller value, and the reassignment horizon is $[\pi_{\text{start}}, \pi_{\text{end}}]$, where $\pi_{\text{start}} = \pi_s + \partial$ and $\pi_{\text{end}} = \max\{\pi_d{}^i | \, i \in \{1, \ldots, N_n\}\}$. The assignment of tasks in D_n within $[\pi_{\text{start}}, \pi_{\text{end}}]$ will affect some originally assigned tasks in the same horizon, denoted by $D_o = \{j \in D \,|\, E_j \in (\pi_{\text{start}}, \pi_{\text{end}})\}$. Hence, tasks in $D_v = D_o \cup D_n$ need to be reassigned.

Before reassignment, we need to compute the "initial location" of each UAV as well as the maximal number of tasks and flying distances that an UAV can perform within $[\pi_{\text{start}}, \pi_{\text{end}}]$. Specifically, if k-UAV is performing j-task at π_{start}, it is available only after this task is completed, and its initial location is exactly j-task's point. Otherwise, since v_k remains constant, k-UAV's location can be calculated based on the task points and the flight time before π_{start}, and then the remaining flying distance is updated accordingly. Moreover, k-UAV can perform up to N^k tasks, where $N^k = N_{\text{lim}}{}^k - N_s{}^k - N_e{}^k$ and $N_s{}^k$ (resp. $N_e{}^k$) denotes the number of tasks assigned to k-UAV before π_{start} (resp. after π_{end}).

Next, all tasks in D_v are reassigned by DHEDA, in which individuals that do not satisfy time windows are regenerated. After time ∂ is passed, the optimal individual Δ^v is adopted, then each k-UAV can be assigned a corresponding task sequence $\theta_k{}^v$. The original assignment of k-UAV after π_{end} is retained and denoted by $\theta_k{}^e$, so the reassignment task sequence $\theta_k{}^r$ is

$$\theta_k^r = \theta_k^v \oplus \theta_k^e \tag{25}$$

where $\oplus$ means combining two sequences. Hence, the reassignment solution $\boldsymbol{\theta}^r = \{\theta_k{}^r | \, k \in U\}$ is available.

PROPOSITION 3 The obtained solution $\boldsymbol{\theta}^r$ according to (25) is deadlock-free.

PROOF Let Δ be the original assignment's individual and its transition sequence is τ_Δ. We have $\tau_\Delta{}^e \subseteq \tau_\Delta$ and $\Delta^e \subseteq \Delta$, which correspond to the tasks after π_{end} in the original assignment. For solution $\boldsymbol{\theta}^r$ with individual $\Delta^r = \Delta^v \oplus \Delta^e$, we construct the amender $(N, M_0)\|\Delta^r$ by removing some transitions and places whose corresponding tasks do not exist in Δ^r. According to Δ^v, marking M_v indicating that all tasks before π_{end} are completed can be reached from M_0 through $\tau_\Delta{}^v$ (i.e., $M_0[\tau_\Delta{}^v > M_v$). As the remaining transitions can be fired along $\tau_\Delta{}^e$, M_E is reached (i.e., $M_v[\tau_\Delta{}^e > M_E$). Since individual Δ^r whose transition sequence $\tau_\Delta{}^r = \tau_\Delta{}^v \oplus \tau_\Delta{}^e$ satisfies $M_0[\tau_\Delta{}^r > M_E$, $\boldsymbol{\theta}^r$ is deadlock-free. ♣

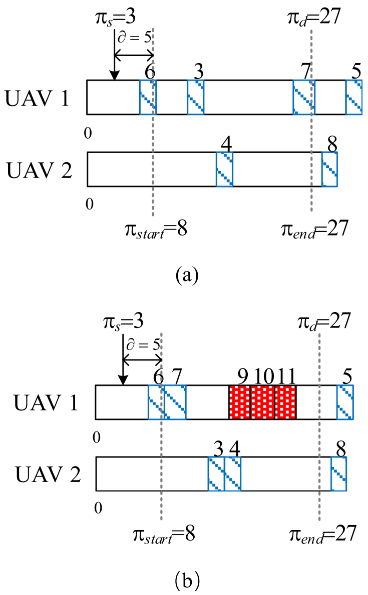
Fig. 6(a) shows the original assignment $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$, where $\theta_1 = <6, 3, 7, 5>$ and $\theta_2 = <4, 8>$, task execution process is marked in blue. A time-sensitive target with time window [3, 27] is detected and the new tasks $D_n = \{9, 10, 11\}$ is marked in red. Then, the disturbed tasks is $D_o = \{3, 4, 7\}$ and $D_v = D_o \cup D_n = \{3, 4, 7, 9, 10, 11\}$. For 1-UAV, we get $\theta_1{}^v = <7, 9, 10, 11>$ after performing DHEDA, the original assignment after $\pi_{\text{end}} = 27$ is $\theta_1{}^e = <5>$, then $\theta_1{}^r = \theta_1{}^v \oplus \theta_1{}^e = <7, 9, 10, 11, 5>$. Similarly, we have $\theta_2{}^r = <3, 4, 8>$. Hence, the reassignment result $\boldsymbol{\theta}^r = \{\theta_1{}^r, \theta_2{}^r\}$ is obtained, as shown in Fig. 6(b).

IV. COMPUTATIONAL EXPERIMENTS

Extensive computational experiments are conducted to describe the performance of the proposed DHEDA.

A. Experimental Setup

For a more comprehensive evaluation, we perform all experiments based on the following three groups of testing instances: small, medium, and large. The number of UAVs,



(a)

(b)

Fig. 6. Gantt chart for task assignment. (a) Original assignment. (b) Reassignment result.

TABLE I
Parameter Size for Each Instance Type

Instance type	u	t	a	$u \times t \times a$
Small	3, 4, 5	2, 3	(200, 3), (400, 4)	3 × 2 × 2 = 12
Medium	10, 12, 14	10, 12	(300, 4), (500, 5)	3 × 2 × 2 = 12
Large	30, 40, 50	30, 40	(400, 5), (600, 6)	3 × 2 × 2 = 12

the number of targets, and the ability limits ($L_{\text{lim}}{}^k$ and $N_{\text{lim}}{}^k$) of k-UAV are represented by u, t, and a, respectively. Table I shows the size of parameters u, t, and a for each instance type, and there are $3 \times (3 \times 2 \times 2) = 36$ combinations. Each combination consists of ten different instances. The positions of UAVs and targets are randomly generated in the range [1, 100], and the j-task's performing time H_j (resp. k-UAV's flying speed v_k) is randomly generated within [1, 3] (resp. [4, 6]).

A comparison is performed with three existing algorithms, which are MTWPS [8], GA [28], and classical EDA (using PDAM to solve deadlocks but without the dedicated local exploitation method). Two graph-based deadlock amending methods, the novel digraph-based method (NDM) and TPG, are used in MTWPS and GA, respectively. These algorithms are measured by the relative percentage value (RPV)

$$\text{PV} = (\text{FV}_a - \text{FV}_b)/\text{FV}_b \tag{26}$$

where FV_a is the fitness of an algorithm for a test instance, and FV_b is the best fitness obtained by all algorithms for the same instance.

To reduce randomness, each algorithm is run 20 times independently, and then, the best RPV (bRPV) and average RPV (aRPV) are used as the performance metrics to evaluate each algorithm. The maximum number of iterations is set as $T_{\max} = 2000$. All algorithms are coded in MATLAB 2021a and run on a PC with an Intel Core i9-9900K CPU @3.60 GHz and 32 GB of RAM in the 64-bit Windows 10 operating system.

TABLE II
Deadlock Detection and Amending Results of PDAM

Scale u×t	Solution amount	Deadlock solution amount	Success rate	running time	Single solution amending time
{4×2}	250	139	100%	0.04	3.13e–4
	500	299	100%	0.07	2.34e–4
	1000	605	100%	0.13	2.30e–4
	2000	1171	100%	0.27	2.37e–4
{12×10}	250	225	100%	0.12	5.70e–4
	500	453	100%	0.22	5.07e–4
	1000	882	100%	0.37	4.30e–4
	2000	1770	100%	0.90	5.13e–4
{40×30}	250	224	100%	0.32	0.0015
	500	451	100%	0.72	0.0016
	1000	901	100%	1.44	0.0016
	2000	1830	100%	2.82	0.0015
{200×200}	250	249	100%	7.84	0.0315
	500	500	100%	28.08	0.0562
	1000	999	100%	43.15	0.0432
	2000	1998	100%	93.39	0.0467

B. Deadlock Amending Verification

Two experiments are conducted to verify the progressiveness of PDAM on deadlock amending. First, we apply PDAM to the individual CMTAP of different scales and compare its performance with NDM and TPG methods. PDAM is established based on PNs, but NDM and TPG resolve deadlocks using the graph formed by CMTAP solutions.

Four combinations of $u \times t$ (i.e., 4×2, 12×10, 40×30, 200×200) are considered in the first experiment, and 250, 500, 1000, 2000 solutions are randomly generated for each combination. We perform PDAM on each individual and record the success rate as well as the running time. Table II shows that as the solution size grows, the probability of deadlock increases accordingly, which can be described by the ratio of deadlock solutions. For large-scale CMTAP, deadlocks arise in nearly all generated individuals. The success rate of PDAM in obtaining deadlock-free solutions is 100% for each instance type, and the running time increases marginally with the problem scale. Even for the solutions of CMTAP with the size $u \times t = 200 \times 200$, PDAM can resolve deadlocks in less than 0.06 s. This result verifies that the computational complexity of PDAM is polynomial from the experimental viewpoint.

In the second experiment, PDAM is compared with NDM and TPG, and six combinations of $u \times t = 4{\times}2$, $12{\times}10$, $40{\times}30$, $100{\times}100$, $200{\times}200$, and $400{\times}400$ are used. We randomly generate 500 individuals for each combination and run three methods on each of them 20 times independently.

TABLE III
Deadlock Detection and Amending Results of PDAM

Scale $u\times t$	Amount	Deadlock solution	PDAM		NDM		TPG	
			Average success rate	Average amending time	Average success rate	Average amending time	Average success rate	Average amending time
{4×2}	500	303	100%	**0.048**	100%	0.605	100%	0.276
{12×10}	500	451	100%	**0.173**	100%	3.654	100%	1.321
{40×30}	500	460	100%	**0.608**	100%	15.229	100%	4.384
{100×100}	500	499	100%	**2.897**	100%	201.576	100%	48.671
{200×200}	500	499	100%	**24.502**	100%	981.621	100%	209.982
{400×400}	500	500	100%	**81.218**	100%	4599.698	100%	468.798

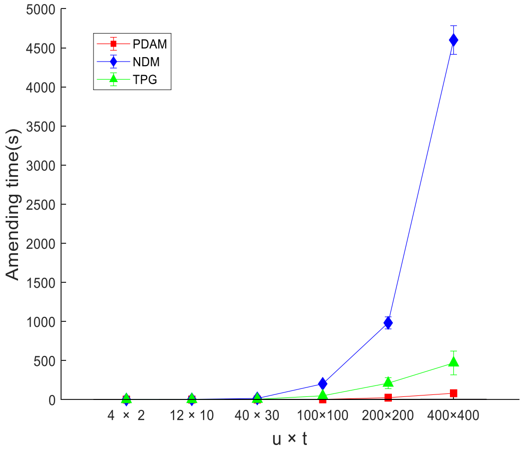


Fig. 7. Variance plots of the amending time for PDAM, NDM, and TPG.

TABLE IV
Factor Levels for Each Parameter

Factor level	N	δ	P_{trans}	P_{shift}	P_{reorg}	ζ
1	20	0.2	0.4	0.2	0.4	7
2	30	0.3	0.6	0.3	0.5	9
3	40	0.4	0.7	0.4	0.6	10
4	50	0.5	0.9	0.5	0.7	12

TABLE V
Orthogonal Array $L_{16}(4^6)$ and aRPV Results or Response Value

Trial	Factor Level						Response Value (*aRPV*)		
	N	δ	P_{trans}	P_{shift}	P_{reorg}	ζ	Small	Medium	Large
1	1	1	1	1	1	1	0.5192	0.2686	0.1966
2	1	2	2	2	2	2	1.0591	0.1545	0.2144
3	1	3	3	3	3	3	2.0027	0.2606	0.2632
4	1	4	4	4	4	4	1.3085	0.5509	0.5616
5	2	1	1	2	2	3	0.9472	0.2042	0.2369
6	2	2	2	1	1	4	0.1015	0.3776	0.2857
7	2	3	3	4	4	1	0.6963	0.4770	0.4409
8	2	4	4	3	3	2	0.2807	0.3909	0.3804
9	3	1	2	3	4	1	0.1416	0.4824	0.4203
10	3	2	1	4	3	2	0.1015	0.5731	0.4355
11	3	3	4	1	2	3	0.2704	0.2633	0.4948
12	3	4	3	2	1	4	0.0657	0.5679	0.5752
13	4	1	2	4	3	3	0.1140	0.2020	0.3678
14	4	2	1	3	4	4	0.3545	0.5131	0.5587
15	4	3	4	2	1	1	0.1647	0.5781	0.5820
16	4	4	3	1	2	2	0.0807	0.6055	0.5629

Table III shows that these three methods can successfully solve all deadlock solutions. However, from the perspective of running time, PDAM performs the best, followed by TPG, and the worst is NDM. Also, as shown in Fig. 7, PDAM has a negligible growth in amending time as the scale grows, while the other two methods exhibit a marked ascending trend. Importantly, PDAM performs well for large-scale CMTAP. For example, when applied to the solutions with $u \times t = 400{\times}400$, the running time of PDAM is decreased by 98.23% ((4599.69−81.21)/4599.69) and 82.67% ((468.79–81.21)/468.79) compared with NDM and TPG, respectively.

C. Parameter Calibration

Six relevant parameters (N, δ, P_{trans}, P_{shift}, P_{reorg}, and ζ) in DHEDA are calibrated by the design of experiment (DOE) [37]. Unlike the traditional approach [38], we perform DOE in the following two steps: we first pick the appropriate factor levels for each parameter by experiments and then execute DOE to obtain the most suitable parameters for each CMTAP instance type. The aRPV is used as the performance metric.

We obtain candidate factor levels for a parameter as follows: ten values of the parameter are tested on a given instance, while all other parameters are fixed. Then, four values that lead to the minimum aRPV are selected as the sought factor levels, and the results are shown in Table IV.

Based on the number of parameters and the factor levels of each parameter, the orthogonal array $L_{16}(4^6)$ is used. Because there are three CMTAP instance types, DOE is executed three times for instances with combinations $u \times t \times a = \{3 \times 2 \times (200, 3)\}$, $\{10 \times 10 \times (300, 4)\}$, and $\{30 \times 30 \times (400, 5)\}$, respectively. Table V shows $L_{16}(4^6)$, which consists of 16 different combinations of parameter factor levels. For each combination, DHEDA is performed 10 times independently, and aRPV is used as the response value.

Table VI shows the statistical analysis of the aRPV results obtained in the test. The significance priority of

TABLE VI
Statistical Analysis and Suggested Parameter Values (SPV)

Factor level		N	δ	P_{trans}	P_{shift}	P_{reorg}	ζ
Small	1	1.2224	0.4305	0.4806	**0.2430**	**0.2128**	**0.3805**
	2	0.5064	**0.4041**	**0.3541**	0.5592	0.5894	**0.3805**
	3	**0.1448**	0.7835	0.7113	0.6949	0.6247	0.8336
	4	0.1785	0.4339	0.5061	0.5551	0.6252	0.4576
	Delta	1.0776	0.3794	0.3573	0.4519	0.4125	0.4531
	Rank	1	5	6	3	4	2
	SPV	40	0.3	0.6	0.2	0.4	7
Medium	1	**0.3087**	**0.2893**	0.3898	0.3788	0.4480	0.4515
	2	0.3624	0.4046	**0.3041**	**0.3762**	**0.3069**	0.4310
	3	0.4717	0.3948	0.4778	0.4117	0.3567	**0.2325**
	4	0.4747	0.5288	0.4458	0.4508	0.5058	0.5024
	Delta	0.1660	0.2395	0.1763	0.0746	0.1990	0.2698
	Rank	5	2	4	6	3	1
	SPV	20	0.2	0.6	0.3	0.5	10
Large	1	**0.3090**	**0.3054**	0.3569	**0.3850**	0.4099	0.4099
	2	0.3360	0.3736	**0.3221**	0.4021	0.3773	0.3983
	3	0.4815	0.4452	0.4606	0.4056	**0.3617**	**0.3407**
	4	0.5178	0.5200	0.5047	0.4515	0.4954	0.4953
	Delta	0.2089	0.2146	0.1827	0.0665	0.1337	0.1546
	Rank	2	1	3	6	5	4
	SPV	20	0.2	0.6	0.2	0.6	10

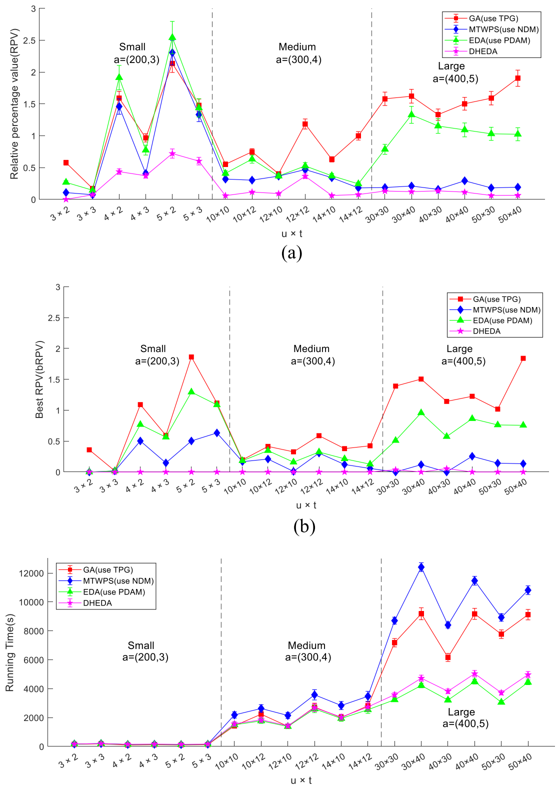
parameters for DHEDA is different for the three CMTAP instance types. Specifically, for small-type instances, N is the most important parameter, ζ ranks second, P_{shift} ranks third, P_{reorg} ranks fourth, δ ranks fifth, and P_{trans} ranks last. For medium-type instances, ζ is the most important parameter, followed by δ, P_{reorg}, P_{trans}, N, and P_{shift}. For large-type instances, δ ranks first, followed by N, P_{trans}, ζ, P_{reorg}, and P_{shift}. Thus, the parameter sets for the remaining experiments are selected as follows: $\{N, \delta, P_{\text{trans}}, P_{\text{shift}}, P_{\text{reorg}}\} = \{40, 0.3, 0.6, 0.2, 0.4, 7\}$, $\{20, 0.2, 0.6, 0.3, 0.5, 10\}$, and $\{20, 0.2, 0.6, 0.2, 0.6, 10\}$ for small, medium, and large type instances, respectively.

D. Comparison With Existing Algorithms

The proposed DHEDA is compared with MTWPS, GA, and EDA and the CMTAP instances described in Section IV-A are used in the comparison.

The comparison results are shown in Table VII, where the performances of the four algorithms vary markedly in terms of aRPV, bRPV, and running time; the optimal values are bolded in black. Fig. 8 plots a set of comparison results and shows that DHEDA obtains the best bRPV and aRPV in nearly all instances, and only in a few cases does MTWPS perform the best.

Fig. 8(c) shows the running time of all algorithms. For small-type instances, deadlocks arise with a low probability, and the GA has the shortest running time due to its simplicity. For large-scale instances, the proportion of deadlock solutions increases, as shown in the experiments in Section IV-B; thus, if the graph-based TPG or NDM is used, it takes a lot of time to resolve deadlocks. Thus, GA (using TPG) and MTWPS (using NDM) take much more time than EDA and DHEDA, which use the proposed PDAM to amend deadlocks. Although DHEDA takes marginally more time than EDA, DHEDA always obtains a better solution because DHEDA uses a dedicated local exploitation method to improve the solution quality, but EDA does not. In general, adding a small additional time cost to achieve a better solution is worthwhile. Therefore, we conclude that DHEDA outperforms existing methods when solving CMTAP, particularly for large instances.



(a)

(b)

(c)

Fig. 8. Plots for GA (use TPG), MTWPS (use NDM), EDA, DHEDA. (a) Variance plot of RPV. (b) Line chart of bRPV. (c) Variance plot of running time.

E. Validation of the Reassignment Method

We verify the proposed match-up-based reassignment method by comparing it with the traditional approach of reassigning all tasks [40]. There are six types of instances with size $u \times t \in \{3\times2, 5\times3, 10\times10, 14\times12, 30\times30, 50\times40\}$ and a is (400, 4). For each instance type, after obtaining the solution by DHEDA, 10 reassignments are performed for a randomly generated time-sensitive target with a time window $[\pi_s, \pi_d]$ such that $\pi_d - \pi_s \in [20, 25]$. The replanning time ∂ is set to one-third of $\pi_d - \pi_s$. We record the reassignment results of two methods.

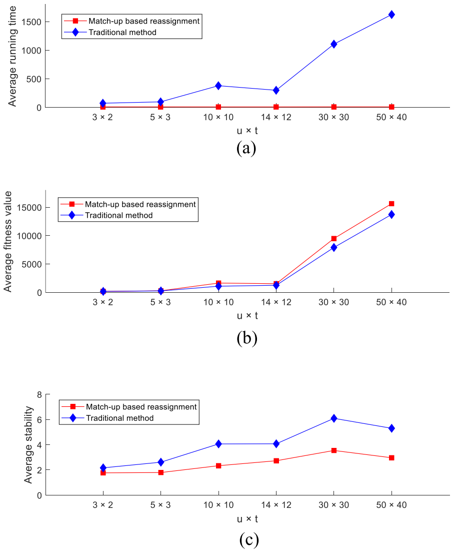
Stability is another crucial factor to assess the reassignment results, because it indicates the time difference between the new assignment and the original one. We can denote the stability parameter as

$$\text{stability}(\theta^{\text{re}}, \theta^{\text{ori}}) = \frac{\sum_{j\in D^{\text{ori}}} \left|E_j^{\text{re}} - E_j^{\text{ori}}\right|}{\left|D^{\text{ori}}\right|} \tag{27}$$

where $\boldsymbol{\theta}^{\text{ori}}$ and $\boldsymbol{\theta}^{\text{re}}$ are the original and new assignments, respectively, D^{ori} is the original task set, E_j^{ori} (resp. E_j^{re}) is the j-task's execution time in the original (resp. new) assignments.

TABLE VII
Comparison Results of Different Algorithms

Instance type	Scale		GA (use TPG)			MTWPS (use NDM)			EDA			DHEDA		
	$u{\times}t$	a	bRPV	aRPV	Time	bRPV	aRPV	Time	bRPV	aRPV	Time	bRPV	aRPV	Time
Small	{3, 2}	(200,3)	0.3591	0.5768	**151.83**	**0**	0.1069	169.14	**0**	0.2675	163.85	**0**	**0**	170.99
	{3, 2}	(400,4)	0.3591	0.4285	**91.00**	**0**	0.1692	110.41	**0**	0.3201	121.86	**0**	**0**	184.26
	{3, 3}	(200,3)	0.0174	0.1714	**181.35**	0.0174	0.0737	200.75	0.0174	0.1437	193.32	**0**	**0.0728**	197.99
	{3, 3}	(400,4)	0.1250	0.2107	**126.67**	**0**	0.1033	155.87	0.0902	0.1475	176.11	**0**	**0.0665**	202.59
	{4, 2}	(200,3)	1.0912	1.5910	**100.20**	0.5025	1.4592	137.95	0.7683	1.9125	136.53	**0**	**0.4359**	148.86
	{4, 2}	(400,4)	3.0912	4.6780	**94.23**	**0**	2.1643	111.977	0.3954	2.1863	150.48	**0**	**0.2180**	190.91
	{4, 3}	(200,3)	0.5935	0.9660	**121.47**	0.1473	0.4065	152.75	0.5643	0.7742	143.75	**0**	**0.3716**	163.78
	{4, 3}	(400,4)	0.6097	0.9301	**127.37**	0.3168	0.6167	166.38	0.3462	0.8053	190.44	**0**	**0.4593**	213.19
	{5, 2}	(200,3)	1.8627	2.1357	**103.52**	0.5025	2.4044	131.92	1.2938	2.5410	130.70	**0**	**0.7203**	145.19
	{5, 2}	(400,4)	7.1357	7.3389	**96.58**	0	1.8761	127.41	1.1669	3.0174	148.35	**0**	**1.4738**	183.55
	{5, 3}	(200,3)	1.1165	1.4780	**125.15**	0.6328	1.3321	154.70	1.0874	1.4371	149.53	**0**	**0.6012**	162.73
	{5, 3}	(400,4)	1.1476	1.4894	**120.78**	0.0702	0.4563	156.27	1.1311	1.4429	203.60	**0**	**0.3525**	226.23
Medium	{10, 10}	(300,4)	0.1969	0.5517	**1430.37**	0.1674	0.3191	2179.63	0.1828	0.4085	1498.45	**0**	**0.0591**	1562.93
	{10, 10}	(500,5)	0.2930	0.7028	**758.96**	0.1099	0.2915	2206.2	0.2125	0.4843	797.22	**0**	**0.0427**	1148.1
	{10, 12}	(300,4)	0.4143	0.7473	2230.79	0.2093	0.3021	2626.91	0.3463	0.6324	**1768.45**	**0**	**0.1154**	1837.65
	{10, 12}	(500,5)	0.4907	0.7290	1197.81	0.1798	0.3852	2659.3	0.2814	0.4967	**969.91**	**0**	**0.0889**	1227.7
	{12, 10}	(300,4)	0.3264	0.4020	1390.58	0.0095	0.3659	2141.73	0.1596	0.3668	**1381.83**	**0**	**0.0920**	1425.57
	{12, 10}	(500,5)	0.3922	0.9872	**763.68**	0.3364	0.4973	2116.2	0.3909	0.6807	797.55	**0**	**0.1184**	1072.9
	{12, 12}	(300,4)	0.5880	1.1841	2722.92	0.3062	0.4691	3573.29	0.3233	0.5256	**2598.01**	**0**	**0.3633**	2682.03
	{12, 12}	(500,5)	0.4583	0.6198	1265.14	0.0168	0.1239	3412.8	0.1668	0.4099	**1256.8**	**0**	**0.0326**	1541.6
	{14, 10}	(300,4)	0.3785	0.6293	1997.27	0.1214	0.3411	2832.07	0.2135	0.3712	**1936.45**	**0**	**0.0615**	2029.73
	{14, 10}	(500,5)	0.7512	0.9404	1320.10	0.1162	0.1795	2773.0	0.4534	0.6804	**1076.7**	**0**	**0.0520**	1390.4
	{14, 12}	(300,4)	0.4243	0.9984	2813.03	0.0573	0.1808	3468.75	0.1267	0.2436	**2538.46**	**0**	**0.0764**	2735.14
	{14, 12}	(500,5)	0.4837	0.8966	1539.04	0.1501	0.2601	3286.0	0.4374	0.7523	**1259.2**	**0**	**0.0993**	1666.5
Large	{30, 30}	(400,5)	1.3910	1.5795	7177.31	**0**	0.1864	8704.87	0.5094	0.7868	**3228.14**	0.0367	**0.1312**	3576.97
	{30, 30}	(600,6)	1.0486	1.4643	5235.3	0.0840	0.1466	7654.5	0.7378	1.0325	**2753.6**	**0**	**0.0624**	3210.7
	{30, 40}	(400,5)	1.5048	1.6202	9181.38	0.1185	0.2102	12404.97	0.9573	1.3276	**4215.67**	**0**	**0.1218**	4721.52
	{30, 40}	(600,6)	1.4984	1.5981	6030.4	**0**	0.1123	10693.1	1.0300	1.4891	**3486.2**	0.0423	**0.0911**	3867.6
	{40, 30}	(400,5)	1.1429	1.3309	6158.57	**0**	0.1575	8401.13	0.5737	1.1524	**3206.14**	0.0557	**0.1337**	3815.59
	{40, 30}	(600,6)	0.9217	1.3665	4975.6	0.0980	0.1704	7438.9	0.6496	0.8659	**2747.5**	**0**	**0.0827**	3077.8
	{40, 40}	(400,5)	1.2260	1.5019	9173.62	0.2546	0.2921	11465.92	0.8634	1.0916	**4484.36**	**0**	**0.1149**	5034.21
	{40, 40}	(600,6)	1.6546	1.9669	6067.8	0.2041	0.2280	9673.8	0.9340	1.6492	**3448.5**	**0**	**0.0728**	3985.3
	{50, 30}	(400,5)	1.0182	1.5894	7767.95	0.1429	0.1808	8925.73	0.7618	1.0298	**3051.64**	**0**	**0.0621**	3714.02
	{50, 30}	(600,6)	1.1646	1.2594	4618.1	**0**	**0.1027**	7049.4	0.6322	0.7750	**2696.8**	0.0328	0.1095	3135.1
	{50, 40}	(400,5)	1.8405	1.9030	9119.69	0.1337	0.1912	10809.72	0.7559	1.0230	**4436.70**	**0**	**0.0644**	4964.68
	{50, 40}	(600,6)	1.9751	2.1413	5806.1	0.1357	0.1892	9194.3	0.7499	1.0057	**3365.4**	**0**	**0.0413**	3875.6



(a)

(b)

(c)

Fig. 9. Comparison of reassignment results. (a) Average running time. (b) Average fitness value. (c) Average stability.

Fig. 9 shows the statistical results of three metrics, including the running time, fitness, and stability. Our method takes significantly less time than traditional one in Fig. 9(a) and, thus, can better cope with dynamic environments. Despite the minimal running time, we still obtain deadlock-free solutions with the good performance [see Fig. 9(b)] because the match-up method reduces the number of reassigned tasks, shortens iteration process, and preserves part of the original optimal assignment. This also leads to better stability of our reassignments [see Fig. 9(c)], especially for the large-scale problems of size $u \times t \in \{30{\times}30, 50{\times}40\}$, with an improvement of nearly 40%.

V. CONCLUSION

This study focuses on CMTAP with temporally coupled constraints for minimizing the total equivalent distance of all tasks. Based on the MILP model of CMTAP, we present a PN amender to resolve the arising deadlocks. This amender is constructed based on a candidate solution, and the corresponding deadlock-free solution is equivalent to a feasible transition sequence that can be fired sequentially in the amender. Then, based on these amenders, the PDAM with polynomial computational complexity is proposed to convert a deadlocked solution to a deadlock-free solution.

The resulting PDAM is effective and fast at solving deadlocks. Then, PDAM is embedded into EDA, creating a new algorithm called DHEDA, which can successfully solve the considered CMTAP. To further improve the solution quality, we establish a dedicated local exploitation method. An adaptive operational probability is explored to preserve the local exploitation ability without markedly increasing the computational burden. To deal with the time-sensitive tasks, a match-up-based reassignment method is proposed. Finally, extensive experiments are performed, and results show that:

1) for large-scale CMTAP, PDAM solves deadlocks in candidate solutions significantly faster than the existing graph-based methods;
2) the proposed DHEDA outperforms existing algorithms in [8] and [28] when solving CMTAP and obtains the best results with only a small additional time cost;
3) the match-up-based reassignment method achieves higher stability and efficiency than traditional method [40].

Extending PDAM to manage more realistic scenarios with simultaneous strikes is planned to be investigated in future work.

REFERENCES

[1] J. Scherer, S. Yahyanejad, and S. Hayat, "An autonomous multi-UAV system for search and rescue," in *Proc. 1st Workshop Micro Aerial Veh. Netw., Syst., Appl. Civilian Use*, 2015, pp. 33–38.

[2] E. P. de Freitas, M. Basso, and A. A. S. da Silva, "A distributed task allocation protocol for cooperative multi-UAV search and rescue systems," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2021, pp. 909–917.

[3] S. Park and Y. Choi, "Applications of unmanned aerial vehicles in mining from exploration to reclamation: A review," *Minerals*, vol. 10, no. 8, p. 663, 2020, doi :10.3390/min10080663.

[4] H. Menouar, I. Guvenc, and K. Akkaya, "UAV-enabled intelligent transportation systems for the smart city: Applications and challenges," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 22–28, Mar. 2017.

[5] M. Suresh and D. Ghose, "UAV grouping and coordination tactics for ground attack missions," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 48, no. 1, pp. 673–692, Jan. 2012.

[6] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative timing missions," *J. Guid., Control, Dyn.*, vol. 28, no. 1, pp. 150–161, 2005.

[7] Z. Zhen, D. Xing, and C. Gao, "Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm," *Aerosp. Sci. Technol.*, vol. 76, pp. 402–411, 2018.

[8] Y. Chen, D. Yang, and J. Yu, "Multi-UAV task assignment with parameter and time-sensitive uncertainties using modified two-part wolf pack search algorithm," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 6, pp. 2853–2872, Dec. 2018.

[9] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1495–1512, 2013.

[10] C. Schumacher, P. Chandler, and M. Pachter, "UAV task assignment with timing constraints via mixed-integer linear programming," in *Proc. AIAA 3rd Unmanned Unlimited Tech. Conf., Workshop Exhibit.*, 2004, p. 6410.

[11] L. Evers, A. I. Barros, and H. Monsuur, "Online stochastic UAV mission planning with time windows and time-sensitive targets," *Eur. J. Oper. Res.*, vol. 238, no. 1, pp. 348–362, 2014.

[12] X. Zhao, Q. Zong, B. Tian, B. Zhang, and M. You, "Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning," *Aerosp. Sci. Technol.*, vol. 92, pp. 588–594, 2019.

[13] J. S. McGrew, J. P. How, B. Williams, and N. Roy, "Air-combat strategy using approximate dynamic programming," *J. Guid., Control, Dyn.*, vol. 33, no. 5, pp. 1641–1654, 2010.

[14] Z. Zhou et al., "When mobile crowd sensing meets UAV: Energy-efficient task assignment and route planning," *IEEE Trans. Commun.*, vol. 66, no. 11, pp. 5526–5538, Nov. 2018.

[15] Y. Eun and H. Bang, "Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm," *J. Aircr.*, vol. 46, no. 1, pp. 338–343, 2009.

[16] Z. Jia, J. Yu, X. Ai, X. Xu, and D. Yang, "Cooperative multiple task assignment problem with stochastic velocities and time windows for heterogeneous unmanned aerial vehicles using a genetic algorithm," *Aerosp. Sci. Technol.*, vol. 76, pp. 112–125, 2018.

[17] F. Ye, J. Chen, Y. Tian, and T. Jiang, "Cooperative multiple task assignment of heterogeneous uavs using a modified genetic algorithm with multi-type-gene chromosome encoding strategy," *J. Intell. Robot. Syst.*, vol. 100, no. 2, pp. 615–627, 2020.

[18] S. Fei, C. Yan, and S. Lin-Cheng, "UAV cooperative multi-task assignment based on ant colony algorithm," *Acta Aeronautica et Astronautica Sinica*, vol. 29, no. 5, pp. 188–189, 2008.

[19] S. Gao, J. Wu, and J. Ai, "Multi-UAV reconnaissance task allocation for heterogeneous targets using grouping ant colony optimization algorithm," *Soft Comput.*, vol. 25, no. 10, pp. 7155–7167, 2021.

[20] K. A. Ghamry, M. A. Kamel, and Y. Zhang, "Multiple UAVs in forest fire fighting mission using particle swarm optimization," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2017, pp. 1404–1409.

[21] V. Gonzalez, C. A. Monje, S. Garrido, L. Moreno, and C. Balaguer, "Coverage mission for UAVs using differential evolution and fast marching square methods," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 35, no. 2, pp. 18–29, Feb. 2020.

[22] X. Liang, H. Chen, and J. A. Lozano, "A Boltzmann-based estimation of distribution algorithm for a general resource scheduling model," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 793–806, Dec. 2015.

[23] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[24] W. Wu, N. Cui, W. Shan, and X. Wang, "Distributed task allocation for multiple heterogeneous UAVs based on consensus algorithm and online cooperative strategy," *Aircr. Eng. Aerosp. Technol.*, Vol. 90, no. 9, pp. 1464–1473, 2018.

[25] K. S. Kim, H. Y. Kim, and H. L. Choi, "A bid-based grouping method for communication-efficient decentralized multi-UAV task allocation," *Int. J. Aeronautical Space Sci.*, vol. 21, no. 1, pp. 290–302, 2020.

[26] Z. Jia, J. Yu, X. Ai, X. Xu, and D. Yang, "Cooperative multiple task assignment problem with stochastic velocities and time windows for heterogeneous unmanned aerial vehicles using a genetic algorithm," *Aerosp. Sci. Technol.*, vol. 76, pp. 112–125, 2018.

[27] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-UAV context," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, vol. 4, pp. 3622–3627.

[28] Q. Deng, J. Yu, and Y. Mei, "Deadlock-free consecutive task assignment of multiple heterogeneous unmanned aerial vehicles," *J. Aircr.*, vol. 51, no. 2, pp. 596–605, 2014.

[29] G. Xu, T. Long, Z. Wang, and L. Liu, "Target-bundled genetic algorithm for multi-unmanned aerial vehicle cooperative task assignment considering precedence constraints," *Proc. Inst. Mech. Engineers, Part G: J. Aerosp. Eng.*, vol. 234, no. 3, pp. 760–773, 2020.

[30] F. Ye, J. Chen, Y. Tian, and T. Jiang, "Cooperative task assignment of a heterogeneous multi-UAV system using an adaptive genetic algorithm," *Electronics*, vol. 9, no. 4, p. 687, 2020, doi: 10.3390/electronics9040687.

[31] W. Tian, L. Liu, Q. Wang, and H. Mu, “Cooperative multiple task assignment using cluster method and bidirectional particle swarm optimization,” in *Proc. IEEE 4th Adv. Inf. Manage., Communicates, Electron. Autom. Control Conf.*, vol. 4, 2021, pp. 797–802.

[32] C. Schumacher, P. Chandler, M. Pachter, and L. Pachter, “UAV task assignment with timing constraints,” in *Proc. AIAA Guid., Navigat., Control Conf. Exhibit*, 2003, Art. no. 5664.

[33] T. Murata, “Petri nets: Properties, analysis and applications,” *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[34] M. Hauschild and M. Pelikan, “An introduction and survey of estimation of distribution algorithms,” *Swarm Evol. Comput.*, vol. 1, no. 3, pp. 111–128, 2011.

[35] Y. Zhang and X. Li, “Estimation of distribution algorithm for permutation flow shops with total flowtime minimization,” *Comput. Ind. Eng.*, vol. 60, no. 4, pp. 706–718, 2011.

[36] F. Qiao, Y. Ma, M. Zhou, and Q. Wu, “A novel rescheduling method for dynamic semiconductor manufacturing systems,” *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 50, no. 5, pp. 1679–1689, May 2020.

[37] J. Antony, *Design of Experiments for Engineers and Scientists*. Amsterdam, The Netherlands: Elsevier, 2014.

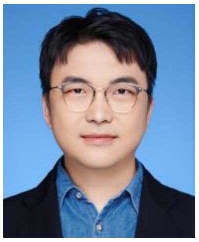
[38] S. Kapsalis, P. Panagiotou, and K. Yakinthos, “CFD-aided optimization of a tactical blended-wing-body UAV platform using the Taguchi method,” *Aerosp. Sci. Technol.*, vol. 108, 2021, Art. no. 106395.

[39] [Online]. Available: https://github.com/ZhangRuiPeng94/TAES-2022-A-Deadlock-free

[40] Y. Wei, M. B. Blake, and G. R. Madey, “An operation-time simulation framework for UAV swarm configuration and mission planning,” *Procedia Comput. Sci.*, vol. 18, pp. 1949–1958, 2013.

Ruipeng Zhang received the B.S. degree in electrical engineering and automation in 2016 from Xi’an Jiaotong University, Xi’an, China, where he is currently working toward the Ph.D. degree in automation science and technology with the School of Automation Science and Engineering.

His research interests include UAV task assignment, decision making for intelligent systems, and satellite edge computing.



Yanxiang Feng received the B.S. and Ph.D. degrees in automation science and technology from Xi’an Jiaotong University, Xi’an, China, in 2010 and 2017, respectively.

He is currently an Associate Professor with the Systems Engineering Institute, School of Automation Science and Engineering, Faculty of Electronic and Information Engineering, Xi’an Jiaotong University. His research interests include Petri net, discrete-event systems, multi-UAV task assignment, and robust control.

Yikang Yang received the B.S., M.S., and Ph.D. degrees in electronic engineering from Xi’an Jiaotong University, Xi’an, China, in 1996, 1999, and 2003, respectively.

He is currently a Professor with the School of Electronic and Information Engineering, Xi’an Jiaotong University. His research interests include satellite navigation, intersatellite link, spacecraft telemetry, tracking, and command (TT&C), and signal processing.

Xiaoling Li received the B.S. degree in electronic science and technology and the Ph.D. degree in control science and engineering from Xi’an Jiaotong University, Xi’an, China, in 2012 and 2019, respectively.

She is currently an Assistant Professor with the School of Electronic and Control Engineering, Chang’an University, Xi’an, China. Her research interests include scheduling of flexible manufacturing systems and discrete event systems.