Robotica (2021), **39**, pp. 2008–2032
doi:10.1017/S0263574721000102

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Distributed task allocation with critical tasks and limited capacity

An Zhang[1,2], Mi Yang[1], Wenhao Bi[1]* and Fei Gao[1]

[1]School of Aeronautics, Northwestern Polytechnical University, Xi'an, China and [2]Key Laboratory of Data Link Technology of 20th Research Institute of CETC, Xi'an, China
*Corresponding author. Email: biwenhao@nwpu.edu.cn

Received: 7 October 2019; **Revised:** 19 January 2021; **Accepted:** 20 January 2021; **First published online:** 8 April 2021

Keywords: Distributed task allocation, Extended performance impact algorithm, Limited capacity, Critical tasks, Multi-UAV system

Abstract

This paper considers the task allocation problem under the requirement that the assignments of some critical tasks must be maximized when the network capacity cannot accommodate all tasks due to the limited capacity for each unmanned aerial vehicle (UAV). To solve this problem, this paper proposes an extended performance impact algorithm with critical tasks (EPIAC) based on the traditional performance impact algorithm. A novel task list resizing phase is developed in EPIAC to deal with the constraint on the limited capacity of each UAV and maximize the assignments of critical tasks. Numerical simulations demonstrate the outstanding performance of EPIAC compared with other algorithms.

1. Introduction

Research into multiple unmanned aerial vehicles (UAVs) has received increasing interest for its capacity to handle difficult and complex missions in various fields including search and rescue, space and underwater exploration, environmental surveillance and monitoring, pick-up, and delivery [1, 2, 3, 4, 5]. Task allocation among a team of homogeneous or heterogeneous UAVs is vital to improve the performance of completing complex missions [6]. For multiassignment problems where each UAV can accomplish a fleet of tasks, a conflict-free and most profitable solution is necessary. However, due to the strong synergies that exist between different tasks, the multiassignment problem is not easy to solve [7]. Hence, the mission can be divided into several subtasks, and an efficient approach is imperative for each UAV to perform a schedule of tasks sequentially to optimize the global objective required by the mission [8].

The task assignment problem considered in this paper is that each UAV can only perform one task at a time, where each task requires only one UAV and each UAV may be assigned to multiple tasks with a compatible type that they can execute based on a schedule [9]. This specified problem is described as a single-task (ST), single-robot (SR), and time-extended assignment (TA) problem [10, 11]. ST-SR-TA problems have been shown to be NP-hard as they are complex and combinatorial decision problems [12]. This paper considers task allocation problems in the scenario where there are critical tasks (e.g., pick-up or delivery of highly valuable materials, surveillance or attack of critical targets), and the capacity of each UAV is fixed due to physical limitations (e.g., fuel, battery, or weapon resource). With the constraints on limited capacities for heterogeneous UAVs, the assignments of these critical tasks should be prioritized due to the peculiar requirement. Moreover, all tasks must be started in their specified time windows, and it is assumed that the UAV starts executing tasks as long as it arrives at the position of the task. The tasks will obtain different scores after being accomplished, which is determined by their start time

and static reward. In conclusion, the main objectives of this task allocation problem of the multi-UAV system in the described scenario are as follows: (1) maximizing the number of assigned critical tasks; (2) maximizing the total score of all assigned critical tasks (critical score); and (3) maximizing the total score of the whole task assignment solution.

To provide a satisfactory solution for the described problem, this paper develops an extended PI algorithm with critical tasks (EPIAC) to achieve the aforementioned objectives. First, the criteria for task inclusion and conflict resolution are modified by using maximizing the total score instead of minimizing the average waiting time of the final assignment solution as the global objective. In addition, a time-discounted reward strategy regarding the time window is defined and incorporated in the global function to guide the solution process in this paper. Finally, to fulfill the constraint on limited capacity and maximize the assignments of critical tasks, a task list resizing phase is proposed. Numerical results demonstrate that the EPIAC can obtain a feasible and conflict-free solution in the described scenario. Moreover, it can provide better performance concerning the total score, critical score, the number of assigned tasks, and critical tasks compared with other algorithms.

The organization of the paper is as follows: The related work is discussed in Section 2. Section 3 presents the definition of problem and introduces the traditional PI algorithm. Section 4 describes the EPIAC proposed in this paper. Simulation results are presented in Section 5, and Section 6 concludes the paper with final remarks.

2. Related work

A considerable part of the literature focuses on specific methods to address the task allocation problem [13]. In past decades, centralized methods have been widely applied to task assignments problems, where a centralized server, such as a ground station or a leader UAV, collects all information of the whole system and generates a conflict-free and optimal solution based on the complete information [14]. The main advantages lie in their ability to obtain a solution that is optimal or very close to it. Much research effort has been done toward extending some traditional centralized approaches, such as ant colony optimization [15], genetic algorithm [16], simulated annealing [17], practical swarm optimization, [18] and artificial neural networks [19]. However, centralized methods may suffer from several weaknesses, especially for large-scale systems. First, the computation time of these algorithms for solving ST-SR-TA problems increases exponentially with the problem size, which leads to imposing a heavy computational burden on the server [20]. Second, the consistent communication between the centralized server and all UAVs is difficult to maintain, while the limited communication ranges for UAVs also reduce the mission coverage. Third, centralized approaches are vulnerable to a single point of failure.

Therefore, particular attention has been paid to the distributed task allocation algorithms, and great efforts have been made for the market-based mechanism [21, 22]. Due to the superiority in terms of robustness and scalability, the auction algorithms have been one of the most popular market-based mechanisms applied in task allocation problems [23, 24], where each UAV calculates the bids for tasks synchronously according to the information from their situational awareness, then the one with the highest bid wins the task and includes the task in its schedule. A specific UAV can be the auctioneer and all bids must be transmitted to the auctioneer in some way that is limited by the network topology.

Recently, the consensus-based bundle algorithm (CBBA) has attracted considerable research interest since it combines the positive properties of the auction and conflict resolution to produce a conflict-free assignment [25]. Unlike traditional auction-based approaches, each UAV individually runs this algorithm without an auctioneer, and a consensus mechanism is used to reach consensus on the winning bids rather than situational awareness. It has been proven that this method can offer similar solutions to some centralized sequential greedy algorithms and 50% optimality is guaranteed [25]. In recent years, several modifications regarding CBBA have been proposed to extend its functions and application areas, such as heterogeneity of UAVs [26, 27], communication constraints [28, 29], dynamic

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

task planning [30, 29], complex cooperation constraints, [31, 32] and others [33, 34, 35]. However, CBBA is based on the auction principles where generally each UAV is selfish in reducing the local cost generated by it. Since a bid in CBBA is calculated according to an extra bundle list for recording the order of adding tasks into its task list, if the inclusion sequence of tasks changes, the bids for the corresponding tasks are no longer valid. Hence, it has to remove all the tasks after the newly added or deleted task in the task list. Inspired by CBBA, Zhao et al. [36] proposed the performance impact (PI) algorithm, which is a distributed heuristic algorithm and directly optimizes the global objective mathematically formulated under the problem of interest. In addition, the PI algorithm places a planner on every UAV and cooperates all UAVs with linked topology and runs concurrently on every UAV by assigning tasks in parallel. It is designed for search and rescue scenarios and aims to minimize the overall waiting time for all survivors. Compared with CBBA, the PI algorithm has been validated to have better performance for solving difficult and time-critical problems [36]. The global objective function can be optimized by iterating the task inclusion phase and conflict resolution phase according to satisfying certain criteria when switching tasks between different UAVs. In recent years, the PI algorithm has been further extended. For example, Whitbrook et al. [37] permitted dynamic online rescheduling and incorporated a soft-max action-selection procedure to improve the exploratory properties of PI. The PI-MaxAssignment was proposed in [38] to maximize task allocations by shifting the assignments to create feasible slots for unassigned tasks. To improve the robustness of PI algorithm, Whitbrook et al. [39] proposed three variants and validated the significant performance of them compared with a robust version of CBBA.

However, there are several drawbacks and challenges for the PI algorithm. First, the solutions are computed based on the assumption that all tasks have identical importance, which means that the priority for some critical tasks is neglected. Second, they cannot assign some specific tasks first when the network capacity of the entire UAV system cannot accommodate all tasks due to the limited capacity. Third, only the deadline for each task is considered in existing PI algorithms, while the specific time window for each task to start is ignored. Finally, all existing PI algorithms are devoted to minimizing the average waiting time for all survivors in search and rescue scenarios. However, maximizing the total global score by executing assigned tasks is the main objective in some specific scenarios, while existing PI algorithms cannot offer a satisfactory solution.

3. Problem description and PI algorithm

This section provides the formal statements of the described task allocation problem and introduces the procedure of PI algorithm.

3.1. Problem description

Consider a scenario with N_u heterogeneous UAVs $\mathbf{V} = [v_1, ..., v_{N_u}]^T$ and N_t tasks $\mathbf{T} = \left[t_1, ..., t_{N_t}\right]^T$, with $N_t > N_u$. An allocation $\mathbf{A} = \left[\mathbf{a}_1, ..., \mathbf{a}_{N_u}\right]^T$ is a part of total task set $\mathbf{T}$ and stores the schedule of each UAV, where $\mathbf{A}$ is ordered by the sequence number of UAVs from 1 to N_u and $\mathbf{a}_i = [t_{i_1}, ..., t_{i_{|\mathbf{a}_i|}}]^T$, $i = 1..., N_u$ is ordered by the arrival time of UAV v_i that arrives at the assigned tasks and is ordered by the actual start time for assigned tasks. The actual size of task list $\mathbf{a}_i$ is decided by the number of tasks assigned to v_i. Since each task can only be assigned to the UAVs with the capability, a compatibility matrix $\mathbf{H}$ with element $h_{i,j}$ is used to define whether UAV v_i is able to perform the task t_j. It is assumed that each UAV is able to autonomously identify the subset of tasks it can perform. $\mathbf{G}(t)$ denotes a symmetric communication matrix, where $g_{i,j} = 1$ indicates that UAV v_i can directly communicate with UAV v_j at time t. It is assumed that every UAV can execute maximum number L_t of tasks; that is, the limited capacity for each UAV is L_t. The described task assignment problem can be written as the following integer program [36]:

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

$$\max \sum_{i}^{N_u} |\mathbf{a}_i^{\mathbf{T}_1}| \tag{1a}$$

$$\max \sum_{i}^{N_u} \sum_{j}^{|\mathbf{a}_i^{\mathbf{T}_1}|} S_{i,j}\left(\mathbf{a}_i^{\mathbf{T}_1}\right) \tag{1b}$$

$$\max \sum_{i}^{N_u} \sum_{j}^{|\mathbf{a}_i|} S_{i,j}(\mathbf{a}_i) \tag{1c}$$

subject to:

$$|\mathbf{a}_i| \leq L_t \tag{2}$$

$$\mathbf{a}_i \in \mathbf{T}, \mathbf{a}_i \bigcap \mathbf{a}_j = \phi \quad \forall i \neq j \tag{3}$$

$$\mathbf{T}_1 \bigcap \mathbf{T}_2 = \phi, \mathbf{T}_1 \bigcup \mathbf{T}_2 = \mathbf{T} \tag{4}$$

$$h_{i,j} = 0, 1 \tag{5}$$

where $\mathbf{a}_i^{\mathbf{T}_1}$ represents the assigned critical tasks in the task list $\mathbf{a}_i$ of UAV v_i. The function $S_{i,j}$ $(\mathbf{a}_i)$ denotes the score obtained by UAV v_i servicing task t_j in its task list $\mathbf{a}_i$. $\mathbf{T}_1$ and $\mathbf{T}_2$ are the critical task set and noncritical task set, respectively. Eqs. (1a), (1b), and (1c) express the three objectives considered in this paper: (1) maximizing the number of assigned critical tasks; (2) maximizing the critical score of the whole solution; and (3) maximizing the total score of the whole solution, respectively. In this paper, the first two objectives are achieved by the newly proposed task list resizing phase. The third objective is used to guide the whole algorithm process, where $S_{i,j}(\mathbf{a}_i)$ defined in the third objective measures the scores obtained by UAV completing tasks and is used to calculate the RPI and IPI in all three phases. Eq. (2) represents that the number of tasks assigned to a UAV must be less than or equal to the limited capacity L_t. Eq. (3) shows that each set of allocations is a subset of all tasks and each task can only be assigned to one UAV. In Eq. (4), it is used to show that the whole task set is divided into two individual subsets, namely, critical tasks set $\mathbf{T}_1$ and noncritical tasks set $\mathbf{T}_2$. Eq. (5) indicates that the element of the compatibility matrix is set to either 0 or 1.

3.2. Score scheme

For the above problem, the following definitions are given:

(1) Time Cost: The time cost τ_{ij} for UAV v_i to execute the task t_j is defined as the predicted time taken by v_i to arrive at the location of the task t_j in its schedule $\mathbf{a}_i$, which includes the duration of earlier tasks in $\mathbf{a}_i$ and travel time to and from those earlier tasks. Actually, the time cost τ_{ij} is the actual start time for UAV v_i to execute task t_j, which is used to calculate the reward of completing the task and constrain the specified time windows.

(2) Time Window: The time window in which the task t_j is allowed to be started is the time interval between the earliest start time τ_{jes} and the latest start time τ_{jls}. If the actual start time of a task is out of the specified time window, this task cannot be allocated. The time window for task t_j is defined as

$$u_j(\tau_{ij}) = \begin{cases} 1, & \tau_{jes} \leq \tau_{ij} \leq \tau_{jls} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

(3) Time-Discounted Reward: The reward a UAV receives from task t_j is related to the time cost τ_{ij} and the static reward R_j, which is defined as

$$R_j^{'}(\tau_{ij}) = e^{-\lambda_j(\tau_{ij}-\tau_{jes})}R_j \tag{7}$$

where τ_{ij}-τ_{jes} is the lag time between the task earliest start time and the UAV actual arrival time. The static reward is defined as the contribution of completely completing the task to the overall goal, and we can also regard the static reward as the inherent importance of the task t_j. To measure the degree of late arrival, a discount parameter $\lambda_j > 0$ is used to penalize task reward. Obviously, the earliest start time τ_{jes} is the best time to start task t_j, and the task reward would be $R_j^{'}(\tau_{ij}) = R_j$ without time discounting.

In conclusion, the score function for a task can be represented as

$$S_{ij}(\tau_{ij}, \mathbf{a}_i) = e^{-\lambda_j(\tau_{ij}-\tau_{jes})}R_j u_j(\tau_{ij}) \tag{8}$$

which reflects the impact of the actual arrival time for UAVs on decreased reward.

3.3. PI algorithm

The PI algorithm is a distributed heuristic task allocation algorithm that runs simultaneously on each UAV in the multi-UAV system [36]. It uses a novel concept called performance impact as the global score function to build task lists iteratively by adding and removing tasks. The removal performance impact (RPI) measures the benefits of removing a task from a task list, while the inclusion performance impact (IPI) measures the benefits of adding a task, and both are incrementally updated by swapping tasks between UAVs as the algorithm proceeds. The PI algorithm considers the total performance impact of task exchanges for all UAVs. Unlike CBBA that targets on optimizing local score, it aims to directly optimize a specified overall objective [38].

Analogous to CBBA, the PI algorithm iterates over a task inclusion phase and a conflict resolution phase. At the beginning, there are two ways to initialize the task lists, winning UAV lists and RPI and IPI lists. The first way is to directly adopt the task assignment results produced by other methods as the initial input. The other way is to initialize all task lists and winning UAVs lists to empty and initialize RPI and IPI lists to 0. During the first phase, each UAV builds a list of tasks that provide the greatest improvement to the global objective determined by the greatest difference between the current global RPI and IPI of a task assignment. Since every UAV builds their task lists locally, two or more UAVs may be assigned to the same task. Therefore, once there are no more tasks to be added or the task lists are full, the second phase, that is, conflict resolution phase, is followed to reach consensus where all UAVs first communicate the information and reach agreement of their current task lists and RPI lists for all tasks with all connected UAVs based on a specified procedure. After that, the UAV with the lowest RPI keeps the conflict task, while others release it through a task removal procedure. The two phases are repeated until a conflict-free task assignment is reached by all UAVs. The main procedure of the PI algorithm is expressed in Algorithm 1.

Regarding the convergence analysis, the PI algorithm is a heuristic distributed algorithm essentially working on the iteration optimization principle (as also used by other centralized optimization methods, for example, genetic algorithms and particle swarm optimization) where each UAV aims to decrease the overall cost at each iteration [36]. In detail, each UAV tries to reduce the overall cost as much as possible by recursively adding/removing tasks into/from the proper position of its task list based on the received RPIs. The RPI and IPI of a task are highly correlated with other tasks in the task list, and they should be updated to reflect the current allocation. Hence, the PI algorithm converges to a feasible and conflict-free solution when no changes can be made in the RPI and IPI values in both the task inclusion phase and consensus phase.

Regarding the computational complexity of the PI algorithm on every UAV, in the task inclusion phase, the major computation complexity comes from the calculation of the IPI of each task that is intended to be inserted into the current task list. Since the benefits of including a task in all possible

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

Algorithm 1 PI Main Program

```
1: Define World, UAVs, Tasks, Network Topology
2: for each UAV v_i do
3:    Initialize the task list a_i, winning UAV lists, RPI and IPI lists
4: end for
5: Initialize Timer T ← 1
6: converged ← false
7: while converged is false do
8:    Phase1: Task Inclusion Phase
9:    Phase2: Conflict Resolution Phase
10:     converged ← Check Convergence
11:     T ← T + 1
12: end while
```

positions should be taken into consideration, the update of the scores for the following tasks is much larger. In addition, there are relatively few computations required for the consensus of the winning UAV list and winning RPI list as most operations are simple rule-based logical judgments. Additionally, the computations requested by determining which tasks are intended to be deleted from the current task list can be omitted. Therefore, the major computational complexity of the conflict resolution phase comes from the update of RPI values of the intended removal tasks and the update of the scores for the remaining tasks in the task list.

4. Extended performance impact algorithm for critical tasks (EPIAC)

To extend the PI algorithm to the scenario with limited capacity and time windows, this paper investigates an extended PI algorithm for critical tasks (EPIAC) to maximize the assignments of critical tasks and critical scores. First, a conflict-free assignment based on fictitious capacity is obtained by repeating the task inclusion phase and conflict resolution phase whose architectures are retained from the PI algorithm. However, to maximize the global total score in EPIAC, the ways to measure the IPI and RPI values for tasks have been changed while the criteria to insert and remove tasks are also modified in these two phases. After generating an original task allocation solution, a novel task list resizing phase is developed to address the constraints related to limited capacity and mandatory assignments of critical tasks. In this newly proposed phase, all UAVs that exceed the limited capacity classify tasks in their task lists based on the consensus information obtained in the conflict resolution phase, and some over-capacity tasks are chosen to be deleted based on the Task Selection to Prune mechanism. After that, another mechanism, Priority Task Inclusion Procedure, is followed to allocate the critical tasks removed in the previous mechanism. As a consequence, a new iteration for maximizing the global optimization is triggered to achieve a new conflict-free assignment. In conclusion, EPIAC iterates between the task inclusion phase and the conflict resolution phase, with some occurrences of the task list resizing phase, until a global conflict-free and constraint-fulfilling task assignment solution is agreed upon by all UAVs. The algorithm terminates when the total score cannot be improved and limited capacity is fulfilled. The main procedure of EPIAC is given in Algorithm 2, where T_1 represents the iteration number for phase 1 and phase 2 to reach convergence, and T_2 denotes the iteration number for phase 3 to fulfill the constraint on limited capacity. Subsection 4.1 introduces some basic definitions of EPIAC, and three phases are described in Subsections 4.2, 4.3, and 4.4.

4.1. Basic definitions of EPIAC

There are two types of local performance impact values used to test the impact of adding and removing tasks on the global cost. It should be noted that the inclusion and removal of a task will have

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

Algorithm 2 EPIAC Main Program

```
Define World, UAVs, Tasks, Network Topology
for each UAV v_i do
    Initialize a_i, γ_i, γ_i^*, β_i, z_i, ϖ_i, s_i
end for
Initialize Timer T_1 ← 1 and T_2 ← 1
overcapacity ← true
while overcapacity is true do
    converged ← false
    while converged is false do
        Phase1: Task Inclusion Phase
        Phase2: Conflict Resolution Phase
        converged ← Check Convergence
        T_1 ← T_1 + 1
    end while
    Phase3: Task List Resizing Phase
    overcapacity ← Check Overload
    T_2 ← T_2 + 1
end while
```

corresponding impacts on the start executing time and score of following tasks, while earlier tasks in the task list are not affected. With the objective of maximizing the score, the RPI of task t_k in $\mathbf{a}_i$ is measured by the difference in score (with and without the removed task) of performing the removed task and all subsequent tasks. It represents the contribution of a task to the local score generated by a UAV. Given a task t_k assigned to UAV v_i, the RPI is defined as

$$w_k(\mathbf{a}_i \ominus t_k) = S_{i,b}(\mathbf{a}_i) + \sum_{z=b+1}^{|\mathbf{a}_i|} \left(S_{i,z}(\mathbf{a}_i) - S_{i,z}(\mathbf{a}_i \ominus t_k)\right) \tag{9}$$

where $\mathbf{a}_i \ominus t_k$ denotes the removal of task t_k from the task list $\mathbf{a}_i$ of UAV v_i, and b symbolizes the position of task t_k in the task list, that is, $a_{i,b} = t_k$. $S_{i,b}(\mathbf{a}_i)$ is the score of the task at position b in task list $\mathbf{a}_i$. The RPI of each task for UAV v_i is stored in an RPI list $\boldsymbol{\gamma}_i = \left[\gamma_{i,1}, ..., \gamma_{i,N_t}\right], i = 1, ..., N_u$ where $\gamma_{i,k}$ represents $w_k(\mathbf{a}_i \ominus t_k)$ for simplicity. To facilitate consensus, define a winning UAV list $\boldsymbol{\beta}_i = \left[\beta_{i,1}, ..., \beta_{i,N_t}\right], i = 1, ..., N_u$ to represent which UAV is assigned to which task in v_i's local view. A global consensus is reached when all UAVs have an identical copy of these two lists.

The IPI is used to measure the newly added task's local performance impact generated by the UAV. The starting time for the tasks after the newly inserted task in the task list could be delayed, which would lead to the decrease on score for the following tasks. As a result, the IPI $w_k^*(\mathbf{a}_i \oplus t_k)$ of task t_k to UAV v_i is the greatest difference in score (with and without the added task) of the added task and subsequent tasks for all possible insert positions and is defined as

$$w_k^*(\mathbf{a}_i \oplus t_k) = \max_{l=1}^{|\mathbf{a}_i|+1} \left\{w_{k,i,l}^{\Delta}\right\} \tag{10}$$

$$w_{k,i,l}^{\Delta} = S_{i,z}(\mathbf{a}_i \oplus_l t_k) + \sum_{z=l}^{|\mathbf{a}_i|} \left(S_{i,z+1}(\mathbf{a}_i \oplus_l t_k) - S_{i,z}(\mathbf{a}_i)\right) \tag{11}$$

where $\mathbf{a}_i \oplus_l t_k$ denotes adding the task t_k into the task list $\mathbf{a}_i$ of UAV v_i at the lth position. The $w_{k,i,l}^{\Delta}$ is calculated as the performance impact value of all possible positions in the task list, and the maximum is chosen as $w_k^*(\mathbf{a}_i \oplus t_k)$. When the task has been included in the task list or does not satisfy the constraints,

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

the IPI is set to 0. An IPI list $\boldsymbol{\gamma}_i^* = \left[\gamma_{i,1}^*, ..., \gamma_{i,N_t}^*\right]$, $i = 1, ..., N_u$ is constructed to store the IPI of each task for UAV v_i, where $\gamma_{i,k}^*$ represents $w_k^*(\mathbf{a}_i \oplus t_k)$ for simplicity.

4.2. Phase 1: Task inclusion phase

During the task inclusion phase, each UAV selects the compatible tasks freely to add to their task lists until no more tasks can be included. The pseudocode of this phase is shown in Algorithm 4. First, the algorithm computes the IPIs for all candidate tasks according to Eqs. (10) and (11) (lines 3–11), where candidate tasks are those compatible and not already in $\mathbf{a}_i$. The RPIs of candidate tasks are initialized as 0 if unassigned or a specific value it received in the conflict resolution phase. Due to the 0 value of initialized RPIs for all tasks, the RPIs must be greater than 0 once assigned. If the IPI of task t_q in $\mathbf{a}_i$ is higher than t_q's RPI in another UAV's task list $\mathbf{a}_j$, the global score can be improved by t_q reallocating to v_i. Based on the new definitions of RPI and IPI, it should be decided which task can be inserted into the current task list $\mathbf{a}_i$ of UAV v_i according to Eq. (12) (line 12).

$$g = \max_{k=1}^{N_t} \left\{\gamma_{i,k}^* - \gamma_{i,k}\right\} \tag{12}$$

If $g > 0$, the task with the greatest difference of IPI and RPI in current candidate task list, that is, the task that can provide the greatest objective improvement, should be selected to add to $\mathbf{a}_i$. The corresponding task $t_g = \arg\max_{k=1}^{N_t} \left\{\gamma_{i,k}^* - \gamma_{i,k}\right\}$ is inserted into the ordered task list $\mathbf{a}_i$ at the position $l_{t_g}^i = \arg w_{i,l}^*\left(a_i \oplus t_g\right)$ (line 14), and the RPI of task t_g is updated using the IPI of task t_g, that is, $\gamma_{i,l}$=$\gamma_{i,l}^*$ (line 15). If $g \leq 0$, IPIs of all tasks are equal to or lower than the current RPIs, which means that the current assignment cannot be improved or the constraints on time windows cannot be met. In this case, the task inclusion process ends. The information of RPI list $\boldsymbol{\gamma}_i$ and UAV list $\boldsymbol{\beta}_i$ are updated at the end of the task inclusion phase (line 16). By iteratively computing the task IPI from Eqs. (10) and (11) and satisfying the criterion (12), the above process continues. The process stops when the criterion is no longer met or the maximum number of N_t tasks has been included in the task list. It is noted that the constraint on limited capacity of each UAV defined in (2) is neglected in this phase; that is, it is assumed that each UAV can include up to N_t tasks. Since each UAV only includes tasks it can execute, the IPI of tasks that cannot be performed will always be set to 0.

4.3. Phase 2: Conflict resolution phase

Because every UAV locally builds its own task list, two or more UAVs may include the same task. To this end, a conflict resolution phase is demonstrated in Algorithm 3. During this phase, some fundamental information should first be shared among all neighboring UAVs where a communication link exists between them based on a network topology. There are five information vectors that should be communicated (line 2), including winning RPI lists $\boldsymbol{\gamma}_i$, winning UAV lists $\boldsymbol{\beta}_i$, second winning RPI lists $\mathbf{z}_i$, second winning UAV lists $\boldsymbol{\varpi}_i$ and time stamps $\mathbf{s}_i$. Every UAV transmits $\mathbf{z}_i$ and $\boldsymbol{\varpi}_i$ across the UAV network with the same procedure used for $\boldsymbol{\gamma}_i$ and $\boldsymbol{\beta}_i$. The $\mathbf{z}_i$ stores the second winning RPI values for each task and $\boldsymbol{\varpi}_i$ contains the corresponding second winning UAVs. The iteration numbers of the last information update from each of the other UAVs are stored in $\mathbf{s}_i$. In the process of transmitting information, each UAV is able to determine the operation (update, leave and reset) by evaluating five vectors when receiving the information from its neighbors [27]. The fundamental idea of this consensus procedure is the following: the received information is compared with the local winning RPI value for a task; if the received RPI is larger than the local winning RPI, then the received one becomes the new winning RPI and the sender UAV becomes the new winning UAV. At the same time, the second winning RPI is chosen between the local old winning RPI and the received second winning RPI, while the second winning UAV is also updated. At the end of this consensus procedure, the vectors $\boldsymbol{\gamma}_i$ and $\boldsymbol{\beta}_i$ are updated to select tasks that need to be removed, while the vectors $\mathbf{z}_i$ and $\boldsymbol{\varpi}_i$ are updated to be used in the task list resizing phase.

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

Algorithm 3 Phase 1 of EPIAC: Task Inclusion Phase

```
1: for each UAV v_i do
2:   while |a_i| ≤ N_t do
3:     for each tasks t_k in problem do
4:       if UAV v_i and task t_k are compatible then
5:         if task t_k not already in task list a_i then
6:           for each insertion position do
7:             Compute w*_k(a_i ⊕ t_k) from (10) and (11)
8:           end for
9:         end if
10:      end if
11:    end for
12:    Compute g from (12) and l from (10) and (11)
13:    if g > 0 then
14:      Add task yielding max g to task list a_i at position l
15:      Set w_k(a_i ⊖ t_k) = w*_k(a_i ⊕ t_k)
16:      Update the RPI list γ_i and UAV list β_i
17:    end if
18:  end while
19: end for
```

In this way, the aforementioned four important vectors could reach global consensus and the time stamp vector $\mathbf{s}_i$ is also updated (line 3).

After the consensus procedure, two local copies of RPI list $\boldsymbol{\gamma}_i$ and winning UAV list $\boldsymbol{\beta}_i$ on UAV v_i have been updated by communicating with the neighbors and are now ready to determine if any tasks in the task list should be removed. First, the candidate task list $\mathbf{d}_i$ intended to be removed from the task list $\mathbf{a}_i$ is determined by $\mathbf{d}_i = \left\{ \mathbf{a}_i | \, \boldsymbol{\beta}_i(\mathbf{a}_i) \neq v_i \right\}$ (line 4). In detail, each UAV v_i checks the winning UAVs of all assigned tasks in its task list $\boldsymbol{\beta}_i(\mathbf{a}_i)$ according to the updated winning UAV list $\boldsymbol{\beta}_i$. Then, the tasks whose winning UAVs are not UAV v_i itself, that is, $\boldsymbol{\beta}_i(\mathbf{a}_i) \neq v_i$, are regarded as conflicting tasks.

Since a higher RPI implies a better assignment, the UAV with lower RPI for a conflicting task should remove the task from its task list. The updated RPI list $\boldsymbol{\gamma}_i^{\diamond} = \left[\gamma_{i,1}^{\diamond}, ..., \gamma_{i,N_t}^{\diamond}\right]$, $i = 1, ..., N_u$ denotes the RPI values of tasks and is calculated based on the current task list according to (9). The vector $\boldsymbol{\gamma}_i^{\diamond}$ is iteratively compared with the previous RPI list $\boldsymbol{\gamma}_i = \left[\gamma_{i,1}, ..., \gamma_{i,N_t}\right]$,$i = 1, ..., N_u$ that emerged from the initial part of the consensus phase. For all candidate tasks $\mathbf{d}_i$, the following criterion is computed (line 5):

$$h = \max_{k=1}^{|\mathbf{d}_i|} \left\{ \gamma_{i,k} - \gamma_{i,k}^{\diamond} \right\} \tag{13}$$

If $h \geq 0$, then the task with the maximum h is removed from both the task list $\mathbf{a}_i$ and the candidate task list $\mathbf{d}_i$ (line 7). Additionally, the time cost and task score of remaining tasks in $\mathbf{a}_i$ as well as the value of $\boldsymbol{\gamma}_i^{\diamond}$ are recomputed and updated again due to the removal of the task (lines 8–9). For each removed task t_k, values of $\gamma_{i,k}$, $\beta_{i,k}$,$z_{i,k}$ and $\varpi_{i,k}$ are reset, while the time stamps $\mathbf{s}_i$ are updated with the last information UAV v_i received for task t_j. Continuing this removal process until the criterion is no longer satisfied or the $\mathbf{d}_i$ is empty, the remained tasks in $\mathbf{d}_i$ (if they actually exist) are put back into the task list $\mathbf{a}_i$ again.

4.4. Phase 3: Task list resizing phase

After obtaining a conflict-free assignment based on the fictitious capacity by repeating the task inclusion phase and conflict resolution phase, the task list resizing phase (see Algorithm 5) is designed to address the constraints on limited capacity of each UAV and the assignments of critical tasks. First, after

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

Algorithm 4 Phase 2 of EPIAC: Conflict Resolution Phase

1: **for** each UAV v_i **do**
2: Communicate the winning RPI list $\boldsymbol{\gamma}_i$, winning UAV list $\boldsymbol{\beta}_i$, second winning RPI lists $\mathbf{z}_i$ and second winning UAV lists $\boldsymbol{\varpi}_i$ with UAV v_j, where $g_{i,j}(t) = 1; j = 1, ..., N_u; j \neq i$
3: Perform the consensus procedure to update $\boldsymbol{\gamma}_i$, $\boldsymbol{\beta}_i$, $\mathbf{z}_i$ and $\boldsymbol{\varpi}_i$
4: Find candidate tasks for removal $\mathbf{d}_i = \left\{ \mathbf{a}_i | \, \boldsymbol{\beta}_i(\mathbf{a}_i) \neq v_i \right\}$
5: Compute h for tasks in $\mathbf{d}_i$ from (13)
6: **while** $\mathbf{d}_i \neq \phi$ and $h > 0$ **do**
7: Remove the task yielding max h from task list $\mathbf{a}_i$ and candidate list $\mathbf{d}_i$
8: Update scores for the tasks followed after the removed tasks in $\mathbf{a}_i$
9: Update the winning RPI list $\boldsymbol{\gamma}_i^{\diamond}$
10: **end while**
11: **end for**

Algorithm 5 Phase 3 of EPIAC: Task List Resizing Phase

1: **for** each UAV v_i **do**
2: Check the cardinality of the task list for all UAVs based on the updated winning UAV lists $\boldsymbol{\beta}_i$
3: **if** there is a UAV v_k exceeding capability $|\mathbf{a}_k| > L_t$ **then**
4: Determine the total number of tasks to be removed $n_{i,r} = |\mathbf{a}_i| - L_t$
5: **if** $n_{i,r} > 0$ **then**
6: Classify the tasks in $\mathbf{a}_i$ into six subsets according to (14)
7: Determine the involved subsets from subset $\boldsymbol{\zeta}_{i1}$ to $\boldsymbol{\zeta}_{i6}$ until $\sum_{k=1}^{K} |\boldsymbol{\zeta}_{ik}| \geq n_{i,r}$, where $0 < K \leq 6$
8: Determine the number of tasks that should be removed in each involved subset $n_{i,r}^k, k = 1, ..., K$
9: **for** each subset $\boldsymbol{\zeta}_{ik}, k = 1, ..., K$ **do**
10: **if** $n_{i,r}^k > 0$ and $n_{i,r}^k = |\boldsymbol{\zeta}_{ik}|$ **then**
11: UAV v_i removes all tasks in subset $\boldsymbol{\zeta}_{ik}$
12: **else if** $n_{i,r}^k > 0$ and $n_{i,r}^k < |\boldsymbol{\zeta}_{ik}|$ **then**
13: Determine and remove $n_{i,r}^k$ tasks chosen from (15)
14: **end if**
15: **end for**
16: Construct a priority inclusion matrix $\mathbf{P}_i^{2\times|\mathbf{T}_{i,cri}|}$ to store the tasks that are critical and removed as the overloaded tasks
17: Communicate $\mathbf{P}_i^{2\times|\mathbf{T}_{i,cri}|}$ with UAV v_j, where $g_{i,j}(t) = 1; j = 1, ..., N_u; j \neq i$
18: Construct a priority task list $\boldsymbol{\eta}_i$ according to $\mathbf{P}_i^{2\times|\mathbf{T}_{i,cri}|}$
19: Execute the task inclusion phase to allocate the tasks in $\boldsymbol{\eta}_i$
20: **end if**
21: **end if**
22: **end for**

the consensus on winning UAV lists $\boldsymbol{\beta}_i$ has been reached, each UAV uses this updated information to locally evaluate if there is an overloaded UAV whose $|\mathbf{a}_i| > L_t$ (line 2). If all UAVs do not violate the constraints of limited capacity, the algorithm stops and the final solution is obtained. Otherwise, each UAV v_i determines the total number of tasks to be removed according to $n_{i,r} = |\mathbf{a}_i| - L_t$ (line 4). The case of $n_{i,r} < 0$ means that UAV v_i is not an overloaded UAV and does not need to delete any task. If $n_{i,r} > 0$, UAV v_i runs two main mechanisms: Task Selection to Prune and Priority Task Inclusion Procedure to deal with the overloaded tasks. More specifically, the task selection to prune procedure is utilized for overloaded UAVs to remove tasks until Eq. (2) is fulfilled (lines 6–15). Afterward, a priority

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

task inclusion procedure is introduced to let the second winning UAV include the critical tasks pruned in the previous procedure (lines 16–19). The task selection to prune mechanism and priority task inclusion mechanism are demonstrated more clearly in following Subsections 4.4.1 and 4.4.1, respectively.

4.4.1. Task selection to prune

There are two main criteria for selecting the tasks to be removed (lines 6–16). One is obtaining the maximum assignments of critical tasks, and the other is choosing the tasks causing the least penalty in terms of score when swapped between UAVs.

For the first criterion, tasks assigned to overcapacity UAV v_i are classified into six subsets described as follows (line 6):

$$\begin{aligned}
\boldsymbol{\zeta}_{i1} &= \left\{ t_j \in \mathbf{a}_i \text{ and } t_j \in \mathbf{T}_2 | \varpi_{ij} > 0 \text{ and } \left| \mathbf{a}_{\varpi_{ij}} \right| < L_t \right\} \\
\boldsymbol{\zeta}_{i2} &= \left\{ t_j \in \mathbf{a}_i \text{ and } t_j \in \mathbf{T}_2 | \varpi_{ij} > 0 \text{ and } \left| \mathbf{a}_{\varpi_{ij}} \right| \geq L_t \right\} \\
\boldsymbol{\zeta}_{i3} &= \left\{ t_j \in \mathbf{a}_i \text{ and } t_j \in \mathbf{T}_2 | \varpi_{ij} = 0 \right\} \\
\boldsymbol{\zeta}_{i4} &= \left\{ t_j \in \mathbf{a}_i \text{ and } t_j \in \mathbf{T}_1 | \varpi_{ij} > 0 \text{ and } \left| \mathbf{a}_{\varpi_{ij}} \right| < L_t \right\} \\
\boldsymbol{\zeta}_{i5} &= \left\{ t_j \in \mathbf{a}_i \text{ and } t_j \in \mathbf{T}_1 | \varpi_{ij} > 0 \text{ and } \left| \mathbf{a}_{\varpi_{ij}} \right| \geq L_t \right\} \\
\boldsymbol{\zeta}_{i6} &= \left\{ t_j \in \mathbf{a}_i \text{ and } t_j \in \mathbf{T}_1 | \varpi_{ij} = 0 \right\}
\end{aligned} \tag{14}$$

The subsets $\boldsymbol{\zeta}_{i1}$, $\boldsymbol{\zeta}_{i2}$, and $\boldsymbol{\zeta}_{i3}$ are composed of noncritical tasks with a nonfull second winning UAV, a full or overcapacity second winning UAV and no second winning UAV, respectively, while the subsets $\boldsymbol{\zeta}_{i4}$, $\boldsymbol{\zeta}_{i5}$, and $\boldsymbol{\zeta}_{i6}$ are similar to $\boldsymbol{\zeta}_{i1}$, $\boldsymbol{\zeta}_{i2}$, and $\boldsymbol{\zeta}_{i3}$ but for critical tasks. It is noted that $\sum_{k=1}^{6} \boldsymbol{\zeta}_{ik} = \mathbf{a}_i$ and each subset is individual in terms of other subsets.

After the tasks in the task list $\mathbf{a}_i$ have been classified, the involved subsets that tend to be removed are chosen according to the total number of tasks to be removed $n_{i,r}$ and the size of each subset (line 7). In detail, the first subset $\boldsymbol{\zeta}_{i1}$ is the first selection for UAV v_i and then subset $\boldsymbol{\zeta}_{i2}$, and subset $\boldsymbol{\zeta}_{i6}$ is the last one. Furthermore, the sum size of involved subsets must cover the number of tasks need to be deleted, that is, $\sum_{k=1}^{K} |\boldsymbol{\zeta}_{ik}| \geq n_{i,r}$ (line 7). The sequence number of the last involved subset is denoted as K, where $0 < K \leq 6$. Next, the number of tasks that should be removed for each involved subset $n_{i,r}^k$ is determined with $\sum_{k=1}^{K} n_{i,r}^k = n_{i,r}$ (line 8). Obviously, if $n_{i,r}^k = |\boldsymbol{\zeta}_{ik}|$, UAV v_i removes all tasks in subset $\boldsymbol{\zeta}_{ik}$ (lines 10–11), while the algorithm does not remove any tasks in subset $\boldsymbol{\zeta}_{ik}$ if $n_{i,r}^k = 0$. When $n_{i,r}^k > 0$ and $n_{i,r}^k \leq |\boldsymbol{\zeta}_{ik}|$, the UAV v_i must choose $n_{i,r}^k$ tasks from subset $\boldsymbol{\zeta}_{ik}$ with the least penalty reduction, which is referred to in the second criterion.

For example, if $|\mathbf{a}_i|$=10, L_t=6, there are four tasks that need to be removed. It is supposed that the size of subsets $\boldsymbol{\zeta}_{i1}$ to $\boldsymbol{\zeta}_{i6}$ is 2, 0, 3, 3, 2, and 0, respectively. As a result, the subsets $\boldsymbol{\zeta}_{i1}$, $\boldsymbol{\zeta}_{i2}$, and $\boldsymbol{\zeta}_{i3}$ are selected since the total number of tasks in these three subsets is $|\boldsymbol{\zeta}_{i1}| + |\boldsymbol{\zeta}_{i2}| + |\boldsymbol{\zeta}_{i3}| = 5$ and is larger than $n_{i,r} = 4$, which means that the sequence number of the last involved subset K is 3. At the same time, the two tasks in $\boldsymbol{\zeta}_{i1}$ are all selected to be removed with the other two tasks from subset $\boldsymbol{\zeta}_{i3}$. The procedure of choosing two tasks to be removed from the three tasks in subset $\boldsymbol{\zeta}_{i3}$ is described in the second criterion.

The second criterion is used to select a specific number $n_{i,r}^K = |\boldsymbol{\zeta}_{iK}| - L_t$ of tasks in the last selected subset whose $|\boldsymbol{\zeta}_{iK}| > n_{i,r}^K$ based on RPI and IPI, where the tasks causing the least penalty of the score are chosen as

$$q = \min_{k=1}^{|\boldsymbol{\zeta}_{iK}|} \left(\gamma_{i,k} - \gamma_{\varpi_{ik},k}^* \right) \tag{15}$$

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

where $|\boldsymbol{\zeta}_{iK}|$ represents the number of tasks in the last selected subset. $\gamma^*_{\varpi_{ik},k}$ denotes the IPI of the second winning UAV ϖ_{ik} to task t_k, and $\gamma^*_{\varpi_{ik},k}$=0 if there is no second winning UAV, that is, t_k belongs to subset $\boldsymbol{\zeta}_{i3}$ or $\boldsymbol{\zeta}_{i6}$. $q \geq 0$ means that UAV v_i removes task t_k and the second winning UAV ϖ_{ik} includes t_k, which will reduce the total score. Hence, the task with the least score loss $t_q = \min_{k=1}^{|\boldsymbol{\zeta}_{iK}|} \left(\gamma_{i,k} - \gamma^*_{\varpi_{ik},k}\right)$ is selected; then, UAV v_i removes it from its task list $\mathbf{a}_i$, and the values of $\gamma_{i,q}$, $\beta_{i,q}$, $z_{i,q}$, and $\varpi_{i,q}$ are set to 0. At the same time, the task list $\mathbf{a}_i$ and local RPI list $\boldsymbol{\gamma}_i$ should be updated. For each task t_k in subset $\boldsymbol{\zeta}_{iK}$, the RPI of the winning UAV removing task t_k is iteratively compared with the IPI of the second winning UAV including t_k, as shown in Eq. (15). This process stops when $n^K_{i,r}$ tasks are removed from the last subset $\boldsymbol{\zeta}_{iK}$ (lines 12–14). The removed task t_q is not allowed to be inserted into $\mathbf{a}_i$ again and the assignability is set to 0, that is, $h_{i,q} = 0$, which guarantees the convergence of EPIAC. Moreover, a novel priority inclusion matrix $\mathbf{P}_i^{2\times|\mathbf{T}_{i,cri}|}$ is designed for UAV v_i to store the critical tasks removed in this mechanism and their corresponding second winning UAVs' IDs, where $|\mathbf{T}_{i,cri}|$ denotes the amount of critical tasks pruned by UAV v_i in phase 3 (line 16). The first row of $\mathbf{P}_i^{2\times|\mathbf{T}_{i,cri}|}$ stores the task IDs and the second row stores second winning UAV IDs.

4.4.2. Priority task inclusion procedure

After the selected tasks have been removed for all overcapacity UAVs, the matrix $\mathbf{P}_i^{2\times|\mathbf{T}_{i,cri}|}$ of UAV v_i is communicated between all UAVs by network topology, where each UAV receives the priority inclusion information about itself and constructs a priority task list $\boldsymbol{\eta}_i$ (lines 17–18). Then, each UAV first tries to include the tasks in $\boldsymbol{\eta}_i$ to maximize the number of assigned critical tasks, while the limited capacity and score optimization are neglected (line 19). It is noted that the same inclusion method as in phase 1 is used with the difference that the tasks that need to be allocated here are the tasks in $\boldsymbol{\eta}_i$. After that, all UAVs start the normal task inclusion phase to rebuild their task list locally and synchronously, which means that a new cycle of phases 1 and 2 is performed to achieve a new conflict-free assignment after the priority task inclusion procedure is completed. In detail, there are two parts of information communicated between all UAVs, including five vectors ($\boldsymbol{\gamma}_i$, $\boldsymbol{\beta}_i$, $\mathbf{z}_i$, $\boldsymbol{\varpi}_i$, $\mathbf{s}_i$) as well as the priority inclusion matrix $\mathbf{P}_i^{2\times|\mathbf{T}_{i,cri}|}$. In addition, the task inclusion phase is composed of priority task inclusion and normal task inclusion, which is determined to allocate more critical tasks and achieve a higher score, respectively.

4.5. Convergence analysis

As EPIAC inherits the same architecture of the PI algorithm and the modifications in this paper do not change the behavior of the two phases concerning convergence; the convergence property is still valid for EPIAC; that is, the convergence time for phases 1 and 2 to generate a conflict-free solution is finite. Therefore, the convergence of EPIAC is mainly determined by the executions of phase 3.

Now, we will show that the phase 3 can only occur a finite number of times. First, define a dynamic assignability matrix $\mathbf{A}^{(k)} \in \{0, 1\}^{N_u \times N_t}$ with elements $\alpha_{ij}^{(k)}$. In addition, if $b_{ij} \leq c_{ij}$ for all elements and $b_{ij} < c_{ij}$ for at least one element, define the relation between two matrices as $\mathbf{B} < \mathbf{C}$. The algorithm starts with $\mathbf{A}^{(1)} = \mathbf{H}$, and the fundamental idea of EPIAC is to modify the element of the dynamic assignability matrix to 0, that is, fewer pairs of the task and UAV. Therefore, the following part verifies that the condition $\mathbf{A}^{(k+1)} < \mathbf{A}^{(k)}$ is always fulfilled as algorithm proceeds.

After a conflict-free assignment has been achieved by repeating phases 1 and 2, all UAVs execute phase 3 to remove overloaded tasks, and consequently a new iteration takes place to converge to a new solution. It is supposed that the dynamic assignment element is set as $\alpha_{ij}^{(k+1)}$=0 when UAV v_i removes task t_j in phase 3. Moreover, the element h_{ij} of compatibility matrix $\mathbf{H}$ is set to 0, which means that the UAV v_i cannot include task t_j any more. This precaution may remain the searching ability of the approach and prevent tasks to be repeatedly included in and excluded from the same UAV. Hence, as the approach proceeds, the number of UAVs that can include task t_j decreases, which leads to a decrease in conflicts over time. In detail, for noncritical task t_j or critical task $t_j \in \boldsymbol{\zeta}_{i6}$ deleted in phase 3, the element

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

$\alpha_{ij}^{(k+1)}$ is reset to 0 with h_{ij}=0 leading to $\mathbf{A}^{(k+1)} < \mathbf{A}^{(k)}$. If the removed critical task t_j belongs to subset $\boldsymbol{\zeta}_{i4}$ or $\boldsymbol{\zeta}_{i5}$, the second winning UAV v_g for task t_j includes t_j into its task list, with $\alpha_{gj}^{(k+1)}$=1. Since v_g is the currently winning UAV for task t_j, it will not remove t_j in phase 1 with Eq. (12) fulfilled, that is, no UAV with a higher RPI for task t_j. Moreover, unlike the principle in which the CBBA needs to remove all the tasks after the added or deleted task in the task list, the inclusion and removal of a task will not lead to the removal of other tasks in the task list. Hence, t_j will not be released due to the task list truncation in phase 1. Despite this, it is still possible that v_g deletes t_j in phase 3 for $|\mathbf{a}_g| > L_t$, with $\alpha_{gj}^{(k+1)}$=0 and h_{gj}=0. After that, a new second winning UAV $\varpi_{gj} = v_p$ with $p \neq g$ adds t_j into $\mathbf{a}_p$. This allows us to conclude that the condition of the removed critical task t_j assigned to a UAV can arise for a maximum of $N_u - 1$ times for each task, that is, the condition $\mathbf{A}^{(k+1)} < \mathbf{A}^{(k)}$ occurs as the algorithm proceeds. Hence, the maximum number of occurrences for phase 3 is $\sum_{j\in\mathbf{T}_1}\sum_{i=1}^{N_u} h_{ij} + \sum_{j\in\mathbf{T}_2}\sum_{i=1}^{N_u} h_{ij} < \sum_{j\in\mathbf{T}}\sum_{i=1}^{N_u} h_{ij} < N_u \times N_t$ which is finite. Therefore, this method can converge to a feasible solution in a finite time.

4.6. Computational complexity and optimality

In this section, the computational complexity of EPIAC is first analyzed. First, as shown in Ref. [36, 37], the major computational complexity for phases 1 and 2 arises from the calculation of RPI and IPI values and the updated scores for the remaining tasks in the task list. In phase 1, a maximum computational complexity of $(|\mathbf{a}_i| + 1)\ (|\mathbf{a}_i| + 2)\ M_1\sigma/2 + |\mathbf{a}_i|\sigma$ is requested to compute the IPI and update the scores for remaining tasks in order to include a new task into $\mathbf{a}_i$. M_1 denotes the number of tasks that are not assigned to v_i and fulfill the constraints in Eq. (5). Considering phase 2, it is assumed that a total of M_2 tasks are intended to be removed from the task list $\mathbf{a}_i$, and this requires a computational complexity of $|\mathbf{a}_i|M_2\sigma - M_2(M_2 + 1)\sigma/2 + (|\mathbf{a}_i| - 1)\sigma$ maximally, where σ denotes the complexity of computing the score of a task. Obviously, the major computational complexity is dominated by the task inclusion phase, and it is $O\big((m_i - |\mathbf{a}_i|)\ |\mathbf{a}_i|^2 M_1\sigma N_u\big)$ at each iteration of the algorithm, where $m_i - |\mathbf{a}_i| = \sum_{j=1}^{N_t} h_{ij}$ denotes the maximum number of tasks that can be inserted into the task list $\mathbf{a}_i$.

Furthermore, regarding phase 3, there are relatively few computations required for the task selection to prune as most operations are simple rule-based logical judgments and assignments. The simple computations involved in classifying the assigned tasks in the task list into six different subsets and in determining which tasks can cause the least reduction in terms of the total score can be omitted. The priority task inclusion procedure can be incorporated in the normal task inclusion phase since priority tasks can be regarded as normal tasks that are not assigned v_i and fulfill Eq. (5). Therefore, we can conclude that the major computational complexity is still dominated by phase 1, which is $O\Big(\sum_{j=1}^{N_t} h_{ij}|\mathbf{a}_i|^2 M_1\sigma N_u\Big)$ at each iteration.

Then, the optimality of the proposed method is discussed. First, as mentioned in the Introduction, the task allocation problem of multi-UAV systems is described as an ST-SR-TA problem which is NP-hard in complexity theory. As the computation time for obtaining a globally optimal solution of such a problem increases exponentially with the problem size, analytic methods are not suitable for large-scale ST-SR-TA problems and most algorithms for solving ST-SR-TA problems can only provide suboptimal solutions [40]. In general, heuristic algorithms are less complex and find a good enough solution within an acceptable time, although the trade-off is that they often provide suboptimal solutions. Therefore, heuristic methods are employed to speed up the process of task allocation while maintaining an efficient and scalable algorithm [40, 41, 42, 43].

Second, many algorithms are discussed, none of which can provide the optimal solution. Among them, the market-based multirobot (MR) coordination has been applied successfully to the ST-SR-TA problem to find suboptimal solutions efficiently in a distributed fashion, as discussed in [40]. In addition, the auction-based algorithms [44, 45, 46] are another kind of distributed method that has been applied to solve ST-SR-TA problems and have been shown to be able to efficiently produce suboptimal solutions, as they employ greedy-based strategies in their task inclusion phase. This reason also applies to the CBBA [25] since a market-based decision strategy is utilized as the mechanism for the decentralized

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

Table I. *Specific scenario.*

	Reconnaissance	**Attack**
UAV Velocity	30 m/s	50 m/s
Task Duration	300 s	350 s
UAV Start Position	10,000 m × 10,000 m × 0 m ground space	
Task Position	10,000 m × 10,000 m × 1000 m 3-D space	
Task Existence Time	Task Duration × 2	
Time Discounting Factor	0.002	

task inclusion phase. Furthermore, it is suggested that PI algorithm suffers from suboptimality because the solution is prone to becoming trapped in local optima rather than reaching global optimality [36], and its modifications can only provide suboptimal solutions [9, 39, 38, 37].

Although the EPIAC retains the same architectures of the first two phases in the PI algorithm and a new task list resizing phase is developed, the nature of the heuristic has not been changed due to the addition of a new phase. Hence, EPIAC cannot guarantee the optimal solution.

5. Numerical results

To verify the effectiveness of EPIAC, five series of simulations are constructed. The scenario descriptions and simulation settings are introduced in Section 5.1; then, the detailed simulation results for increasing limited capacity, increasing the UAV number and increasing the number of critical tasks are demonstrated in Sections 5.2, 5.3, and 5.4, respectively. Next, the effects of the proposed method under different reward schemes and network topologies are illustrated in Sections 5.5 and 5.6, respectively. Finally, a simple scenario is displayed in Section 5.7 to intuitively show the specific task assignment result. All the algorithms are conducted in MATLAB with an Intel Core i5-7400 CPU @3.00 GHz.

5.1. Scenario and simulation setup

To test the performance of EPIAC, the scenario in [38] is used and is summarized in Table I. Two types of heterogeneous UAVs (reconnaissance UAVs and attack UAVs) are considered, the velocity is set to 30 m/s for reconnaissance UAVs and 50 m/s for attack UAVs, and the velocities remain constant all the time. Likewise, there are two types of tasks being introduced here, namely, reconnaissance tasks and attack tasks, where the reconnaissance tasks can only be performed by reconnaissance UAVs and attack tasks are only available for attack UAVs. It is assumed that the tasks and UAVs are equally split into two types, respectively, and a predefined number of tasks are chosen to be critical arbitrarily. The tasks are randomly generated in a 3-D space spanning $10,000\ m \times 10,000\ m \times 1000\ m$, and the UAVs are randomly placed on the corresponding 2-D ground plane. The allowed time window for each task to be started is set as double task duration time, where the task duration is 300 s for executing a reconnaissance task and 350 s for an attack task. The discounting factor λ is set to 0.002. Given the random initialization parameters of tasks and UAVs, it is possible that some tasks cannot be executed by any UAV before their deadlines. A total of 50 simulations are conducted for each condition in the following section according to different scenarios randomly generated in the specific region.

5.2. Simulation results of increasing limited capacity

Fig. 1 displays the results obtained with N_u=4, N_t=100, and $|\mathbf{T}_1|$=50 where limited capacity L_t ranges from 8 to 18 and the upper bound, which denotes the maximum value for generating the earliest start

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

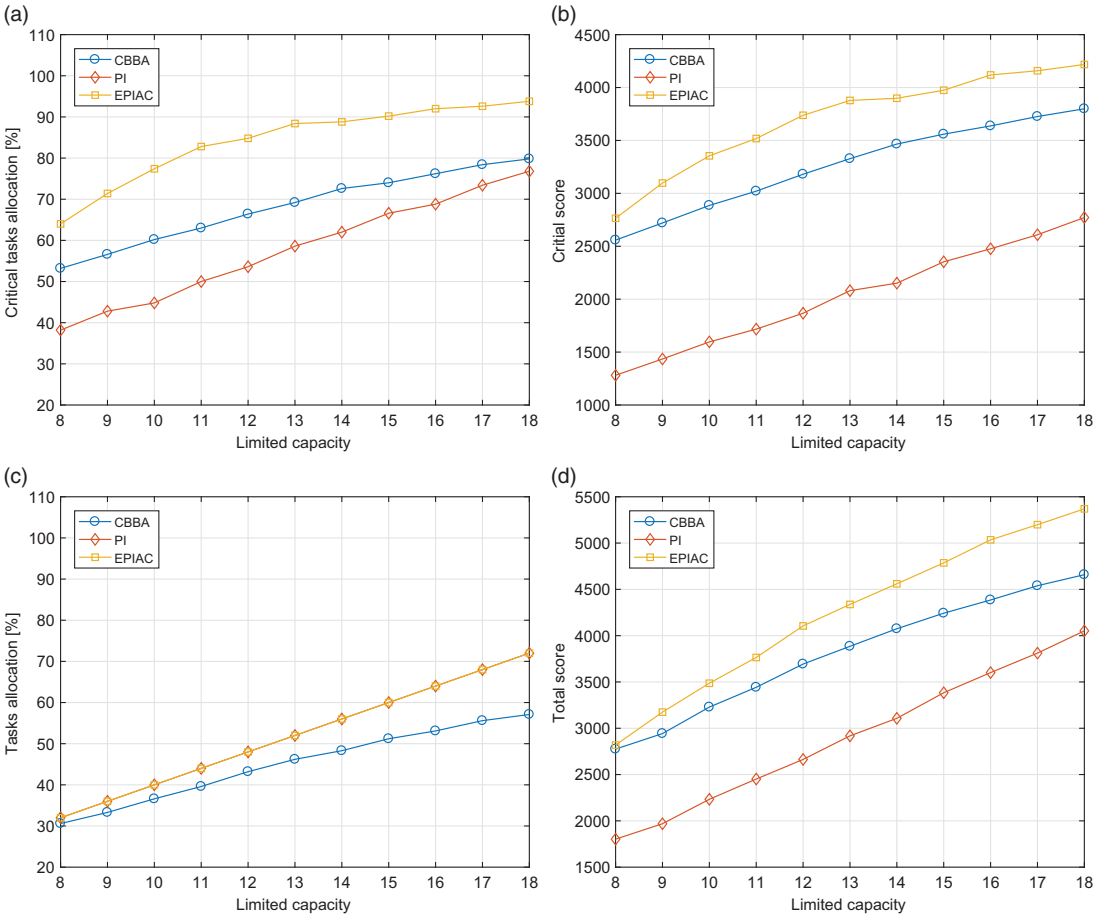

Figure 1. *Analysis of limited capacity effect on (a) critical task assignment percentage, (b) critical score, (c) all tasks assignment percentage, (d) total score for CBBA (blue line, circle marker), PI algorithm (red line, diamond marker), EPIAC (yellow line, square marker). All curves show the mean value over 50 simulations.*

time τ_{jes}, increases from 4000 to 9000 s. The increasing upper bound is used to avoid tight time windows. For example, the upper bound of 4000 s is considerably urgent when the limited capacity is 18 since the durations for two types of tasks are 350 and 300 s, respectively, which means that each UAV can perform up to 13 tasks theoretically with a 4000 s upper bound and that the actual limited capacity is far from the given limited capacity of 18. In this paper, the upper bound is retained from [9, 27]. In addition, the rewards for noncritical tasks are generated between [10-90], while the reward for critical tasks is 100.

Fig. 1(a) indicates that EPIAC always assigns more critical tasks than CBBA and PI algorithm as the limited capacity increases. Moreover, before the limited capacity rises to 13, which is the minimum value for all UAVs to accommodate all critical tasks, the critical task assignment percentage gradually rises to approximately 90% which indicates that it can assign most critical tasks. Additionally, Fig. 1(b) shows that the critical score of the whole assignment increases as the limited capacity increases because UAVs tend to include more critical tasks in the task list. Moreover, EPIAC always achieves a higher critical score than the other two algorithms. The task assignment percentage is displayed in Fig. 1(c) with the result that it increases linearly with the same trend as the PI algorithm, and network capacity is always fully utilized in both algorithms. When $N_u L_t > N_t$, PI, and EPIAC obtain the same solutions all the time since phase 3 is not triggered in EPIAC to remove overloaded tasks. For case $N_u L_t \leq N_t$, the

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

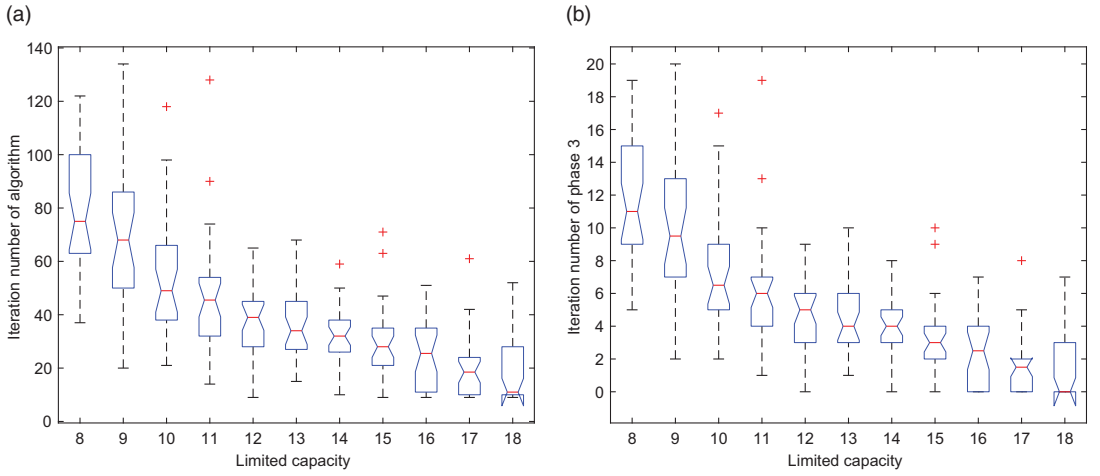

Figure 2. *Iteration analysis of limited capacity effect on EPIAC. All the values are obtained from the average value of 50 independent simulations.*

simulation results show that the PI algorithm can always make full use of its network capacity, which means that the length of task lists for all UAVs are equal to L_t. For EPIAC, a conflict-free task assignment result that ignores the limited capacity constraint is first achieved by phase 1 and 2. Then, after phase 3 is utilized to remove all overloaded tasks, the task lists of all UAVs are equal to L_t. Hence, we conclude that the PI algorithm always has exactly the same task assignment percentage as EPIAC as network capacity increases. In contrast, the performance of CBBA is relatively poor in assigning all tasks, and it cannot fully utilize every UAV's capacity. Because EPIAC can include more critical tasks with higher rewards, its total score is higher than the other two algorithms. Fig. 2(a) and (b) demonstrate the number of iterations of EPIAC and phase 3 alone, respectively. It is noted that the iteration numbers of phase 3 decrease as the limited capacity increases since overloads are less frequent when limited capacity increases. At the same time, because the occurrence of phase 3 will result in a new cycle of phases 1 and 2 to achieve convergence again, EPIAC is able to converge with fewer iterations as there are decreased iterations of phase 3.

5.3. Simulation results of increasing the UAV number

The second series of simulations focuses on the analysis of EPIAC scalability for the number of UAVs. Fig. 3 shows the simulation results obtained with L_t=5, N_t=100, and $|\mathbf{T}_1|$=50. The upper bound of time windows for each task is set to 3000 s, with rewards of critical tasks set to 100 and [10–90] for noncritical tasks. Fig. 3(a) and (b) show that the critical task assignment percentage and the critical score increase with a similar trend as the UAV number improves before the threshold (at approximately 10) which is the minimum value to guarantee the assignments of all critical tasks. After this threshold, that is, the limited capacity is greater than 11, the critical task assignment percentage gradually rises to 100%, which means that all critical tasks can be assigned. It is noted that the critical score still slowly increases when all critical tasks are assigned with the limited capacity. Moreover, the critical score and the critical task assignment percentage of CBBA and PI algorithms are both lower than EPIAC. Fig. 3(c) and (d) indicate that the proposed method can always offer a constraint satisfaction solution to assign all tasks and achieve a higher total score. The iteration numbers of EPIAC and phase 3 are illustrated in Fig. 4(a) and (b), respectively. Since every UAV includes tasks individually and locally, there are more conflicts with increased UAVs, which results in more iterations to reach convergence. As a consequence, the iteration numbers of EPIAC increase as the number of UAVs increases. Moreover, the network capacity grows to accommodate more tasks (including critical tasks); hence, fewer occurrences of phase 3 are

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

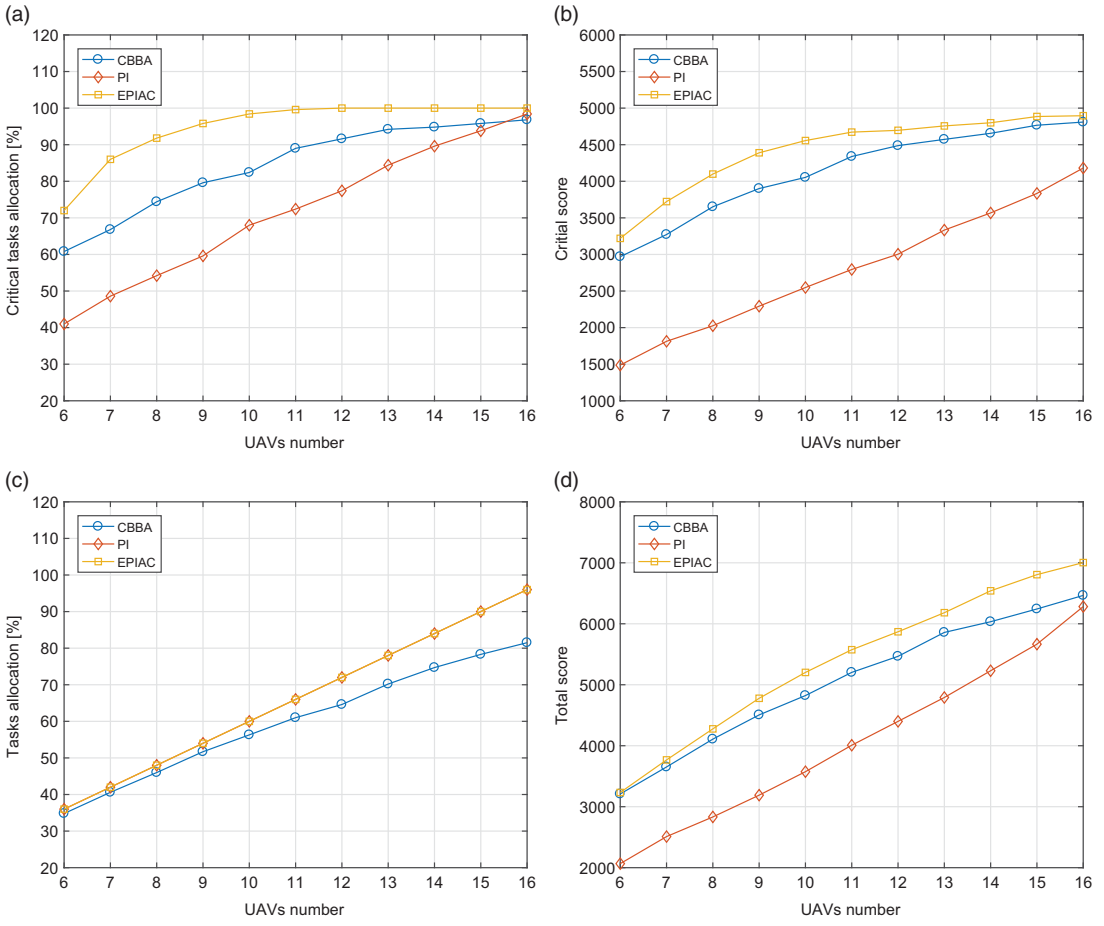

Figure 3. *Analysis of UAV number effect on (a) critical task assignment percentage, (b) critical score, (c) all task assignment percentage, (d) total score for CBBA (blue line, circle marker), PI algorithm (red line, diamond marker), and EPIAC (yellow line, square marker). All curves show the mean value over 50 simulations.*

required to maximize the assignments of critical tasks, which leads to a decrease in the iteration numbers of phase 3 in Fig. 4(b).

5.4. Simulation results of increasing the number of critical tasks

The third series of simulations is conducted to test the performance of EPIAC under different numbers of critical tasks. The results displayed in Figs. 5 and 6 are obtained with N_u=6, N_t=100, and L_t=8 using different numbers of critical tasks that range from 30 to 80 and an upper bound of time windows is set as 6000 s. The rewards for noncritical tasks are generated between [10–90] and the reward of critical tasks is set as 100. Fig. 5(a) demonstrates that the critical task assignment percentage decreases as the number of critical tasks increases, since the network capacity N_uL_t=48 is fixed. Due to the increasing number of critical tasks among all tasks, all three algorithms prefer to include the critical tasks that can be started as early as possible to achieve higher scores. Hence, the critical scores for the three algorithms increase all the time, as shown in Fig. 5(b). Furthermore, Fig. 5(c) and (d) show the results of the overall task allocation percentage and total score. EPIAC and PI can allocate the same number of tasks all the time, while CBBA assigns relatively few tasks. However at the same time, CBBA and EPIAC can achieve

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

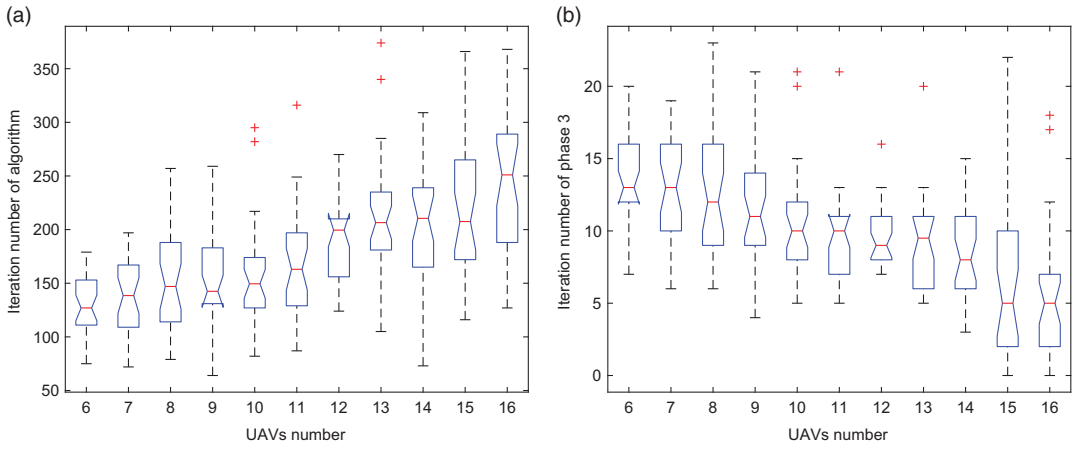

Figure 4. *Iteration analysis of UAV number effect on EPIAC. All the values are obtained from the average value of 50 independent simulations.*

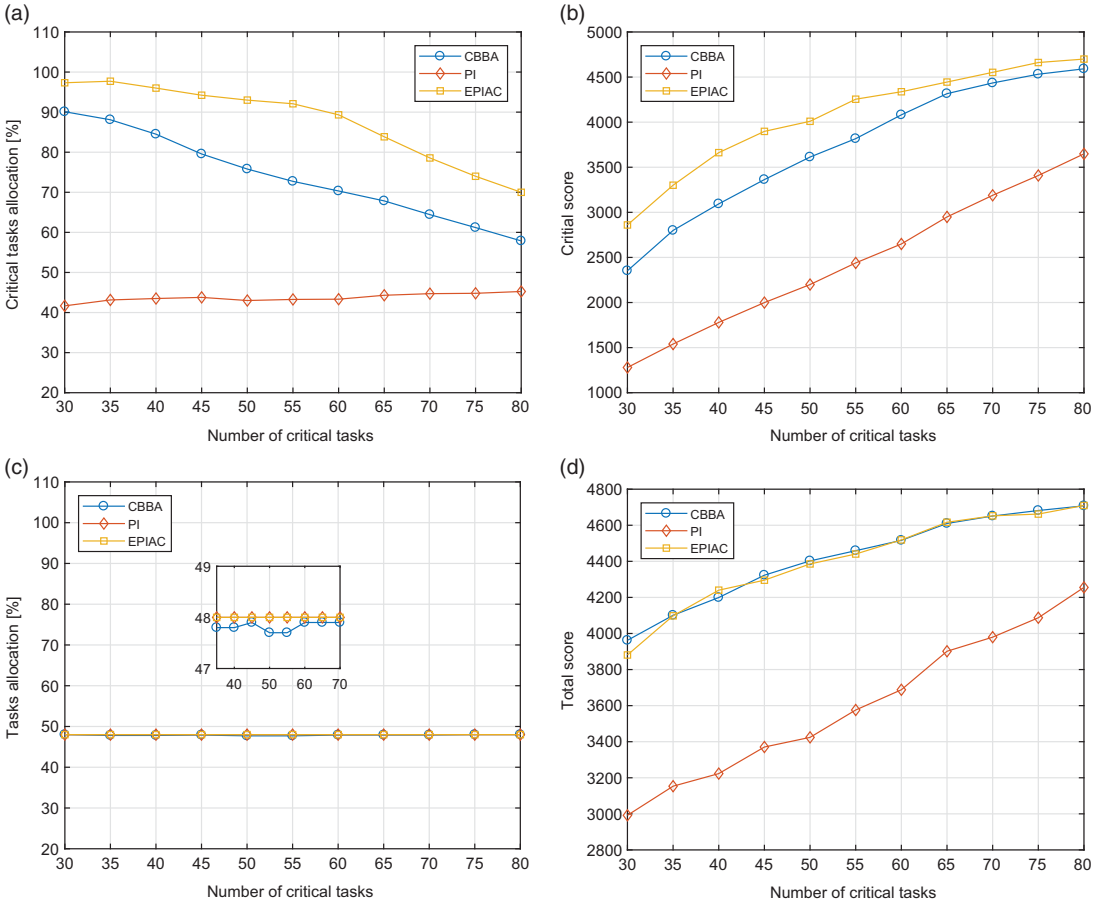

Figure 5. *Analysis of critical task number effect on (a) critical task assignment percentage, (b) critical score, (c) all task assignment percentage, (d) total score for CBBA (blue line, circle marker), PI algorithm (red line, diamond marker), and EPIAC (yellow line, square marker). All curves show the mean value over 50 simulations.*

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

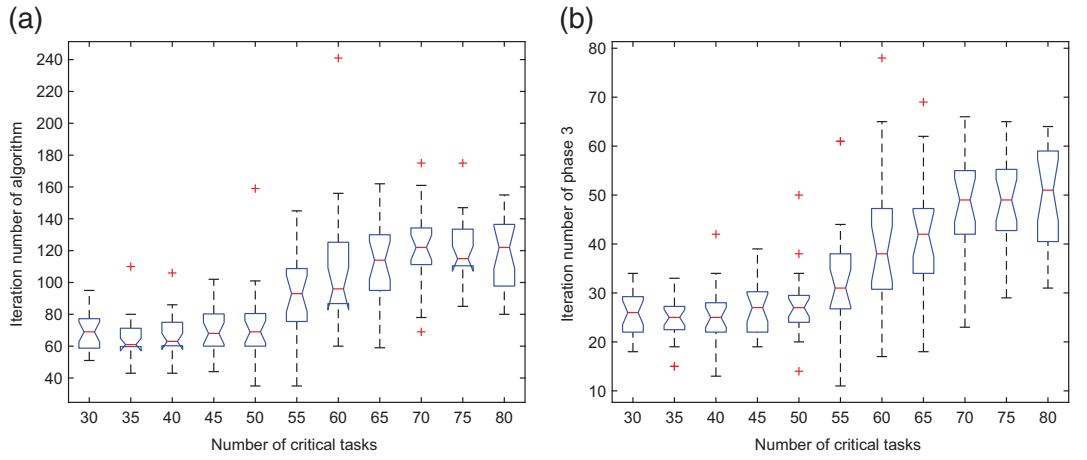

Figure 6. *Iteration analysis of critical task number effect on EPIAC. All the values are obtained from the average value of 50 independent simulations.*

approximately the same total score, and PI has a poor performance, which means that CBBA can obtain a higher score for each task in the task list. It can be concluded that the EPIAC has an outstanding performance with different numbers of critical tasks compared to the other two algorithms. Fig. 6(a) and (b) display the iteration numbers of EPIAC and phase 3, respectively. As the number of critical tasks increases and the network capacity is fixed, more phase three scenarios are triggered to allocate more critical tasks. Therefore, the iteration numbers of the algorithm increase as the iteration numbers of phase 3 increase.

5.5. Simulation results of different rewards

To further verify the significant performance of EPIAC under different reward schemes, two additional different reward schemes for all tasks are utilized. For the results illustrated in Fig. 7(a) and (b), the reward of the first reward scheme for all tasks is randomly generated between [10-100], while all rewards are set to 100 in the second reward scheme displayed in Fig. 7(c) and (d). Four settings are considered including $N_u = 4$ with $L_t = 8$ and 16, $L_t = 6$ with $N_u = 6$ and 12, respectively, for settings 1 to 4. At the same time, the condition $N_t = 100$, $|\mathbf{T}_1| = 50$ always holds. Obviously, as shown in Fig. 7(a) and (c), the critical score of the EPIAC is much higher than that of the CBBA and PI algorithms for both reward schemes. At the same time, CBBA can obtain a higher critical score than PI when the first reward scheme is adopted, which means that CBBA is preferred to allocate more valuable tasks than PI. In addition, the PI and CBBA have a similar performance when all tasks are assigned the same static reward. Furthermore, EPIAC can also assign more critical tasks compared with PI and CBBA, as shown in Fig. 7(b) and (d). When all three algorithms adopt the first reward scheme, a slight improvement in the allocation percentage of critical tasks can be ignored. We can conclude that EPIAC can ignore the effect of different rewards on critical tasks and always allocate more critical tasks than the other two algorithms.

5.6. Simulation results of different network topologies

Since EPIAC is a distributed task allocation algorithm and runs on each UAV concurrently, it is necessary to test the effect of different network topologies on the communication burden. Fig. 8 shows four common topologies. In a mesh topology as in Fig. 8(a), every UAV in the network is connected to all other UAVs, while every UAV only has connections to its two neighbors in a row topology shown in Fig. 8(b). Moreover, the circle topology shown in Fig. 8(c) connects the tail of the row topology. The

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

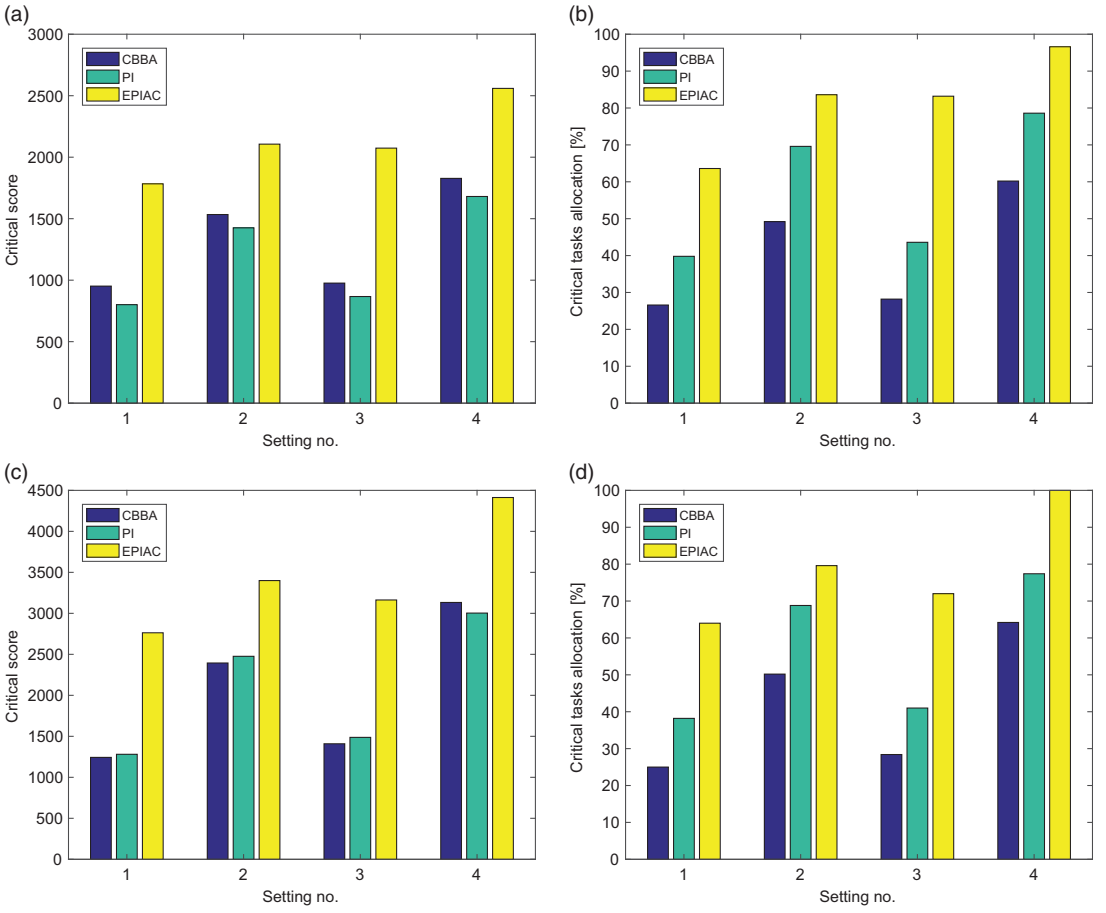

Figure 7. *Analysis of different reward scheme effects on EPIAC for critical score and critical task assignment percentage. In (a) and (b), the reward for all tasks are randomly generated between [10-100], while all rewards are set to 100 in (c) and (d).*

star topology depicted in Fig. 8(d) is extended by row communication with a center UAV that is able to correspond with several other UAVs.

Table II displays the iteration numbers and run time for all four network topologies considering different settings. This paper considers three settings to test the effect of the communication network on the three algorithms with L_t=6, N_t=100, $|\mathbf{T}_1|$=50, and the upper bound is set to 3000 s. There are three conditions where the number of UAVs N_u=6, 12, 18 represents $N_u L_t < |\mathbf{T}_1|$, $|\mathbf{T}_1| < N_u L_t < N_t$, and $N_t < N_u L_t$, respectively. The maximum differences of the iteration numbers and run time for four different communication topologies are also shown in Table II. All results are obtained from the average value of 50 independent simulations.

In the mesh network topology, all UAVs can exchange information directly, so the least iteration numbers are required to reach consensus, as shown in Table II, and most iterations are requested in the star network topology in most cases compared with the other three topologies. The same conclusion holds for the run time of four different network topologies. Furthermore, the results indicate that the EPIAC consumes more iterations and run time to converge to a conflict-free solution than the other two algorithms due to processing the overloaded tasks. Since the allocation of critical tasks is the main focus when considering the constraint on the limited capacity of UAVs, the run time is not the most important factor. Therefore, the run time of EPIAC is acceptable. In addition, the results demonstrate

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

Table II. *Analysis of four network topologies on three algorithms.*

	Network Topology	CBBA			PI			EPIAC		
		$n = 6$	$n = 12$	$n = 18$	$n = 6$	$n = 12$	$n = 18$	$n = 6$	$n = 12$	$n = 18$
Iteration numbers	Mesh	11.94	22.38	31.64	14.70	24.94	52.92	136.00	189.97	63.93
	Row	12.86	28.04	41.36	15.20	29.80	73.14	137.60	242.80	69.80
	Circular	12.86	28.04	41.36	15.20	29.80	73.14	137.40	251.23	69.80
	Star	14.54	30.86	43.34	16.56	30.16	70.24	144.17	252.30	75.17
	max %diff.	21.77%	38.25%	37.03%	12.38%	20.93%	38.21%	5.88%	32.82%	17.58%
Run time	Mesh	0.497	1.618	3.715	0.267	0.884	3.334	1.053	2.703	4.262
	Row	0.484	1.677	3.553	0.274	1.008	3.869	1.386	2.715	4.433
	Circular	0.489	1.657	3.565	0.285	1.009	3.965	1.202	2.803	4.558
	Star	0.503	1.699	3.888	0.282	1.004	3.454	1.177	3.082	4.999
	max %diff.	3.925%	4.767%	9.593%	6.315%	11.95%	15.91%	24.02%	12.29%	14.74%

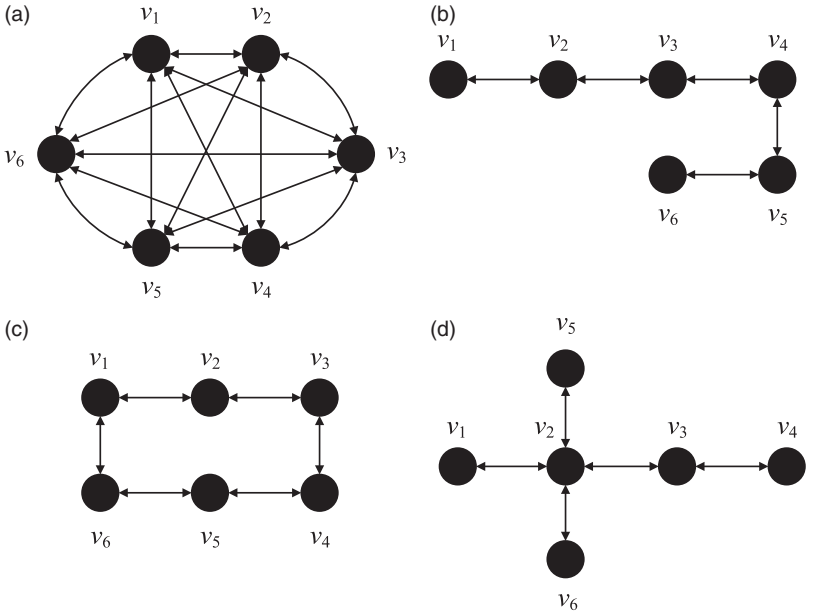

Figure 8. *Four different network topologies among UAVs tested in this paper. (a) Mesh Topology. (b) Row topology. (c) Circular topology. (d) Star topology.*

that the maximum difference percents of iteration numbers for the three algorithms are 38.25%, 38.21%, and 32.82%. The maximum difference percents of run times are 9.593%, 15.91%, and 24.02% for the three algorithms. Therefore, it can be concluded that the effect of the network topology is tolerable.

5.7. A specific task assignment scenario

To intuitively show the results of task assignment for a multi-UAV system, one scenario with all critical tasks assigned is shown in Fig. 9. This scenario involves N_u=6 heterogeneous UAVs (three reconnaissance UAVs and three attack UAVs) and each UAV can perform up to L_t=4 tasks. There are

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

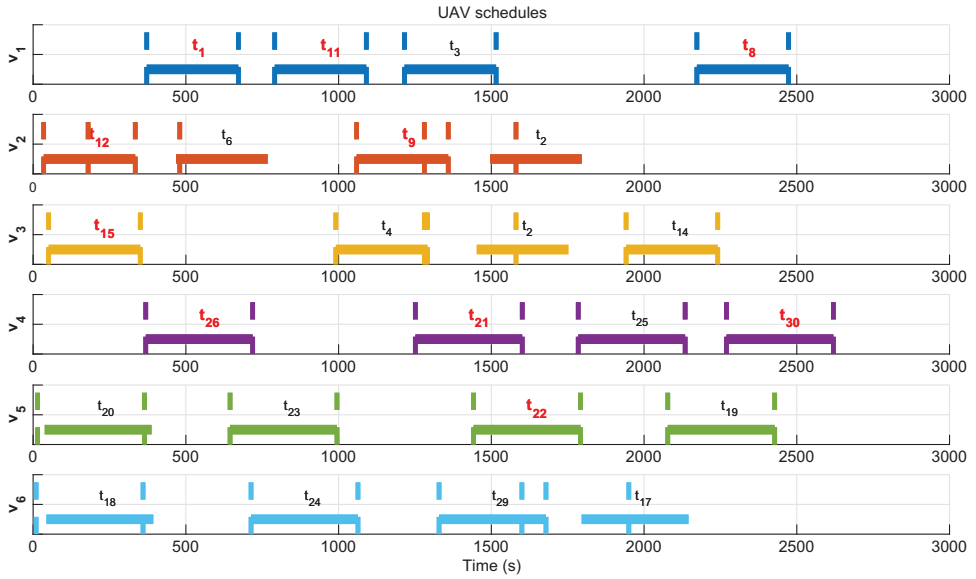

Figure 9. *Each UAV's execution time schedule from EPIAC, where the critical tasks are expressed in bold red font. The actual execution time period for each task is plotted as the horizontal line, and the time window for starting each task is denoted by the vertical lines.*

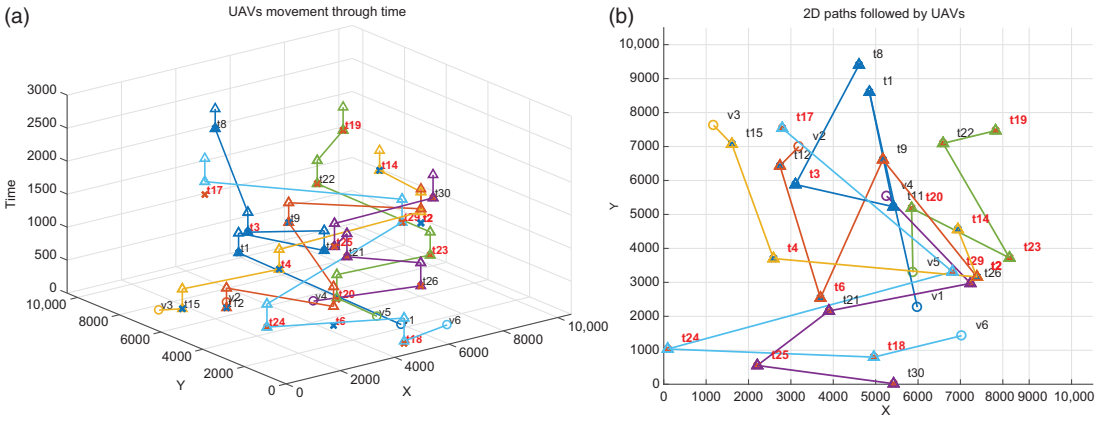

Figure 10. *UAV movements through time. One color represents the path of one UAV. The locations of UAVs are denoted as circles and the tasks as crosses. The triangle indicates the actual start time and finish time of an assigned task.*

N_t=30 tasks in this scenario of which $|\mathbf{T}_1|$=10 are critical tasks. The upper bound to generate the earliest start time of each task is 3000 s. For simplicity, the allowed time window for each task to be started is set as a ST duration.

The overall task assignment results are shown in Fig. 9, where 10 critical tasks are all allocated. It is noted that the actual start time of most tasks is just slightly after their earliest start time to be executed, especially for critical tasks. It means that EPIAC can ensure that each critical task is implemented as early as possible and thus obtain a higher task reward. Fig. 10 shows the UAV movements through time where the locations of UAVs are denoted as circles and crosses represent tasks. In Fig. 10(a), the Z-axis represents time, and the Z-axis coordinate of each allocated task indicates its earliest start time. Each

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

of two triangles represents the actual start time and completion time of each assigned task. It can be seen that all tasks are started after their earliest start time. The two-dimensional paths followed by UAVs projected on the XY-axis are shown in Fig. 10(b).

6. Conclusion

The task allocation of multi-UAV systems is regarded as a very challenging problem, and many research efforts have been made. However, rarely has research considered the task allocation problem under the scenario with constraints on the limited capacity of each UAV and the maximization of the assignments of critical tasks. To this end, this paper proposed an EPIAC, which is a distributed heuristic algorithm, to maximize the assignments of critical tasks with the limited capacity of each UAV and the tight time window for each task. First, the task inclusion phase and conflict resolution phase are used to obtain a no-conflict solution without considering the limited capacity of each UAV. The same architectures of these first two phases are retained from the PI algorithm, while the criteria to insert and remove tasks are modified to maximize the total score of the final solution. Second, the third phase, that is, the task list resizing phase, is proposed to address the constraints on limited capacity for each UAV. In detail, there are two mechanisms composed of the task selection to prune mechanism that is designed to select overloaded tasks to remove, and the priority task inclusion procedure that is used to allocate the critical tasks removed in the task selection to prune mechanism.

Simulations results show that EPIAC offers a significant performance in different conditions (increasing UAV number, increasing limited capacity, and increasing critical task numbers) when not all tasks can be assigned. Although the EPIAC requires more iterations to converge to a feasible and conflict-free solution, it can assign more critical tasks with limited capacity than CBBA and PI algorithms. In addition, it is able to achieve a higher critical score and a total score with significant performance in assigning time-critical tasks.

Future work will focus on further improving the critical task assignment percentage and reducing the iteration numbers to converge to a feasible solution.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (Grant No. 61903305), the Aeronautical Science Foundation of China (No. 201905053001), the Key Laboratory Open Foundation of Data Link Technology (CLDL-20182113), and the Research Funds for Interdisciplinary Subjects, NWPU.

References

[1] A. Khamis, A. Hussein and A. Elmogy, "Multi-Robot Task Allocation: A Review of the State-of-the-Art," **In:** *Cooperative Robots and Sensor Networks 2015* (Springer, 2015) pp. 31–51.
[2] A. Zhang, D. Zhou, M. Yang and P. Yang, "Finite-time formation control for unmanned aerial vehicle swarm system with time-delay and input saturation," *IEEE Access* **7**, 5853–5864 (2018).
[3] W. Shen, L. Wang and Q. Hao, "Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey," *IEEE Trans. Syst. Man Cybern. C (Appl. Rev.)* **36**(4), 563–577 (2006).
[4] Y. Cao, W. Yu, W. Ren and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Inform.* **9**(1), 427–438 (2012).
[5] S. Xie, A. Zhang, W. Bi and Y. Tang, "Multi-UAV mission allocation under constraint," *Appl. Sci.* **9**(11), 2184 (2019).
[6] A. Torreño, E. Onaindia, A. Komenda and M. Štolba, "Cooperative multi-agent planning: A survey," *ACM Comput. Surv. (CSUR)* **50**(6), 84 (2018).
[7] A. M. Khamis, A. M. Elmogy and F. O. Karray, "Complex task allocation in mobile surveillance systems," *J. Intell. Robot. Syst.* **64**(1), 33–55 (2011).
[8] G. A. Kaminka, "Autonomous Agents Research in Robotics: A Report from the Trenches," **In:** 2012 *AAAI Spring Symposium Series* (2012).
[9] X. Chen, P. Zhang, G. Du and F. Li, "A distributed method for dynamic multi-robot task allocation problems with critical time constraints," *Robot. Autonom. Syst.* **118**, 31–46 (2019).
[10] E. Nunes, M. Manner, H. Mitiche and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robot. Autonom. Syst.* **90**, 55–70 (2017).

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

[11] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.* **23**(9), 939–954 (2004).

[12] J. Bruno, E. G. Coffman Jr. and R. Sethi, "Scheduling independent tasks to reduce mean finishing time," *Commun. ACM* **17**(7), 382–387 (1974).

[13] G. A. Korsah, A. Stentz and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Robot. Res.* **32**(12), 1495–1512 (2013).

[14] L. Huang, Y. Ding, M. Zhou, Y. Jin and K. Hao, "Multiple-solution optimization strategy for multirobot task allocation," *IEEE Trans. Syst. Man Cybern. Syst.* (2018).

[15] P. C. Pendharkar, "An ant colony optimization heuristic for constrained task allocation problem," *J. Computat. Sci.* **7**, 37–47 (2015).

[16] I. Younas, F. Kamrani, M. Bashir and J. Schubert, "Efficient genetic algorithms for optimal assignment of tasks to teams of agents," *Neurocomputing* **314**, 409–428 (2018).

[17] O. Catoni, "Solving scheduling problems by simulated annealing," *SIAM J. Cont. Optim.* **36**(5), 1539–1575 (1998).

[18] P.-Y. Yin, S.-S. Yu, P.-P. Wang and Y.-T. Wang, "Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization," *J. Syst. Softw.* **80**(5), 724–735 (2007).

[19] I. Sabuncuoglu and B. Gurgun, " A neural network model for scheduling problems," *Eur. J. Operat. Res.* **93**(2), 288–299 (1996).

[20] D. Bertsimas and R. Weismantel, *Optimization Over Integers*, Volume 13 (Dynamic Ideas Belmont, 2005).

[21] M. B. Dias and A. Stentz, "Opportunistic Optimization for Market-Based Multirobot Control," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE 2002), Volume 3, pp. 2714–2720.

[22] G. Oh, Y. Kim, J. Ahn and H.-L. Choi, "Market-based task assignment for cooperative timing missions in dynamic environments," *J. Intell. Robot. Syst.* **87**(1), 97–123 (2017).

[23] G. Oliver and J. Guerrero, "Auction and Swarm Multi-Robot Task Allocation Algorithms in Real Time Scenarios," **In:** *Multi-Robot Systems, Trends and Development* (IntechOpen, 2011).

[24] D. P. Bertsekas, "The auction algorithm for assignment and other network flow problems: A tutorial," *Interfaces* **20**(4), 133–149 (1990).

[25] H.-L. Choi, L. Brunet and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.* **25**(4), 912–926 (2009).

[26] H.-L. Choi, A. K. Whitten and J. P. How, "Decentralized Task Allocation for Heterogeneous Teams with Cooperation Constraints," **In:** *Proceedings of the 2010 American Control Conference,* (IEEE 2010), pp. 3057–3062.

[27] G. Binetti, D. Naso and B. Turchiano, "Decentralized Task Allocation for Surveillance Systems with Critical Tasks," *Robot. Autonom. Syst.* **61**(12), 1653–1664 (2013).

[28] S. Ponda, J. Redding, H.-L. Choi, J. P. How, M. Vavrina and J. Vian, "Decentralized Planning for Complex Missions with Dynamic Communication Constraints," ***In:*** *Proceedings of the 2010 American Control Conference (IEEE 2010)*, pp. 3998–4003.

[29] L. Johnson, S. Ponda, H.-L. Choi and J. How, "Asynchronous Decentralized Task Allocation for Dynamic Environments," **In:** *Infotech@ Aerospace 2011* (2011), p. 1441.

[30] N. Buckman, H.-L. Choi and J. P. How, "Partial Replanning for Decentralized Dynamic Task Allocation," **In:** *AIAA Scitech 2019 Forum* (2019), p. 0915.

[31] A. K. Whitten, H.-L. Choi, L. B. Johnson and J. P. How, "Decentralized Task Allocation with Coupled Constraints in Complex Missions," ***In:*** *Proceedings of the 2011 American Control Conference, (IEEE 2011)*, pp. 1642–1649.

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

[33] L. B. Johnson, H.-L. Choi, S. S. Ponda and J. P. How, "Decentralized task allocation using local information consistency assumptions," *J. Aerosp. Inform. Syst.,* 103–122 (2017).

[34] S. Hunt, Q. Meng and C. J. Hinde, "An Extension of the Consensus-Based Bundle Algorithm for Multi-Agent Tasks with Task Based Requirements," ***In:*** *2012 11th International Conference on Machine Learning and Applications (IEEE 2012)*, Volume 2, pp. 451–456.

[35] K.-S. Kim, H.-Y. Kim and H.-L. Choi, "Minimizing Communications in Decentralized Greedy Task Allocation," *J. Aerosp. Inform. Syst.,* 1–6 (2019).

[36] W. Zhao, Q. Meng and P. W. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE Trans. Cybern.* **46**(4), 902–915 (2015).

[37] A. Whitbrook, Q. Meng and P. W. Chung, "Reliable, distributed scheduling and rescheduling for time-critical, multiagent systems," *IEEE Trans. Automat. Sci. Eng.* **15**(2), 732–747 (2017).

[38] J. Turner, Q. Meng, G. Schaefer, A. Whitbrook and A. Soltoggio, "Distributed task rescheduling with time constraints for the optimization of total task allocations in a multirobot system," *IEEE Trans. Cybern.* **48**(9), 2583–2597 (2017).

[39] A. Whitbrook, Q. Meng and P. W. Chung, "Addressing robustness in time-critical, distributed, task allocation algorithms," *Appl. Intell.* **49**(1), 1–15 (2019).

[40] M. B. Dias, R. Zlot, N. Kalra and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. IEEE* **94**(7), 1257–1270 (2006).

https://doi.org/10.1017/S0263574721000102 Published online by Cambridge University Press

[41] S.-W. Lin and F. Y. Vincent. "A simulated annealing heuristic for the team orienteering problem with time windows," *Eur. J. Oper. Res.* **217**(1), 94–107 (2012).
[42] Z. Luo, H. Qin and A. Lim, "Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints," *Eur. J. Oper. Res.* **234**(1), 49–60 (2014).
[43] S. E. Butt and T. M. Cavalier, "A heuristic for the multiple tour maximum collection problem," *Comput. Oper. Res.* **21**(1), 101–111 (1994).
[44] J. Guerrero and G. Oliver, "Auction and swarm multi-robot task allocation algorithms in real time scenarios," *Multi-Robot Syst. Trends Dev.* 437–456 (2011).
[45] C. Wei, K. V. Hindriks and C. M. Jonker, "Dynamic task allocation for multi-robot search and retrieval tasks". *Appl. Intell.* **45**(2), 383–401 (2016).
[46] D.-H. Lee, "Resource-based task allocation for multi-robot systems," *Robot. Autonom. Syst.* **103**, 151–161 (2018).

Cite this article: A. Zhang, M. Yang, W. Bi and F. Gao (2021). "Distributed task allocation with critical tasks and limited capacity", Robotica **39**, 2008–2032. https://doi.org/10.1017/S0263574721000102