Optimization of Dynamic Task Allocation forMulti-UAV Systems: Search and Rescue Scenario

Rahim Ali Qamar
Isra University
Mubashar Sarfraz
National University of Modern Languages
Sajjad Ahmed Ghauri
Isra University
Noman Anwar Baig
n.baig@rgu.ac.uk
Robert Gordon University
Tanweer Ahmad Cheema
Isra University

Research Article

Keywords: Multi-UAV Systems, Dynamic Task Allocation, Enhanced Compromised Dynamic Performance Impact, Search and Rescue.

Posted Date: January 25th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3879027/v1

Additional Declarations: No competing interests reported.

Optimization of Dynamic Task Allocation for Multi-UAV Systems: Search and Rescue Scenario

Rahim Ali Qamar, Mubashar Sarfraz, Sajjad A. Ghauri*, Nauman Anwar Baig*, Tanweer Ahmad Cheema

***Abstract*—Unmanned aerial vehicles (UAVs) are becoming increasingly popular due to their versatility and cost-effectiveness and are used in various scenarios, such as sea exploration and search and rescue missions. Urban search and rescue requires a prompt and effective response to unforeseen events like faults in UAVs, changes in task duration and deadline, new UAV arrivals, and task deletions. This paper extends the compromised dynamic performance impact (CDPI) algorithm and introduces an enhanced CDPI (ECDPI) that can effectively handle dynamic events during task execution. For dynamic event handling in the ECDPI algorithm, the communication and conflict resolution phase uses the time cost vector of each UAV for its assigned task, as well as improvements in the detect dynamic activity function. In addition, limitations and cases against dynamic events are discussed to exploit the proposed scheme effectively. The simulation results demonstrate that ECDPI is highly effective at handling dynamic events during the task execution phase.**

***Index Terms*—Multi-UAV Systems, Dynamic Task Allocation, Enhanced Compromised Dynamic Performance Impact, Search and Rescue.**

I. Introduction

In urban areas, unpredictable catastrophic events, including floods, fires, and earthquakes, can occur at random, causing widespread damage and affecting a large population. The need for swift and efficient SAR services on a large scale is of utmost importance to safeguard human life. Unmanned aerial vehicles (UAVs) are considered to be one of the most significant advancements of the 21^{st} century due to their cost-effectiveness, flexibility, and high efficiency [1]–[3]. Therefore, UAVs can be utilized in disastrous situations to support disaster survivors, efficiently contributing to reducing injuries, deaths, and property loss [4]–[6].

Multi-UAV systems have become more popular than single-UAV systems and have emerged as a promising area of research due to their ability to handle hazardous missions in various domains, such as search and rescue operations, underwater and space exploration, environmental monitoring, as well as pick-up and delivery [7]–[11].

Rahim Ali Qamar, Sajjad A. Ghauri & Tanweer Ahmad Cheema are with the School of Engineering & Applied Sciences, ISRA University, Islamabad, Pakistan; rahim_qamar@hotmail.com (R.Q); sajjad.ghauri@iiu.edu.pk(S.A.G) tanweer313@yahoo.com (T.A.C)

Mubashar Sarfraz is with the Faculty of Engineering and Computing, NUML, Islamabad, Pakistan; mubashar.sarfraz@numl.edu.pk (M.S)

Nauman Anwar Baig is with the School of Engineering, Robert Gordon University, Scotland; n.baig@rgu.ac.uk (N.A. B)

* Corresponding Author: Dr. Nauman Anwar Baig & Prof. Dr. Sajjad A. Ghauri; email: n.baig@rgu.ac.uk & dr.saghauri@gmail.com

The authors in [12] introduced the Performance Impact (PI) algorithm, a multi-UAV task allocation algorithm designed for search and rescue scenarios. The PI algorithm uses static task allocation to assign tasks to UAVs, and the tasks are predetermined before the allocation process begins. In the PI algorithm for the SAR scenario, each UAV can handle up to L tasks. Additionally, each task has a deadline, and failing to meet the deadline results in task failure, which is known as a time-limited scenario.

The PI algorithm is extended as a compromised dynamic PI algorithm (CDPI) to handle the dynamic arrival of new tasks during task execution [13]. The authors used the VIKOR algorithm to select a task for allocation, considering parameters such as task cost, priority, deadline, and remaining UAV battery [14]. The PI algorithm does not take into account task priority or battery time limitations. The authors presented a comparison table of task allocation algorithms, including their constraints, application area, and environment. The comparison table serves as a foundation for additional expansion of the algorithms being compared.

The CDPI algorithm can only handle one dynamic event, such as task allocation during execution. SAR is a complex scenario that involves multiple variables that can change during the execution phase. These variables include the UAV battery limit, task duration, task deadline, the arrival of new UAVs, faults in UAVs, changes in task location, and deletion of a task before execution. This paper discusses the problem of task allocation in SAR situations that arise in the context of an urban disaster. The SAR scenario is constantly changing, and the task allocation algorithm for a multi-UAV system needs to be adapted to handle dynamic events such as the emergence of new tasks during execution, changes in battery capacity, and variations in task duration.

In this paper the challenge of task allocation for multiple UAVs is addressed, where each UAV is capable of executing only one task at a time and each task requires the attention of only one UAV. A UAV can handle multiple tasks within its assigned limit, which are executed according to their indices in the task vector. The problem can be described as a single-task robot instantaneous assignment (ST-SR-IA) problem, and these problems are NP-hard due to complex and combinatorial decisions [15].

A. Motivation

Natural disasters such as floods, wildfires, earthquakes, and human-made disasters such as industrial accidents are causing an increasing number of fatalities. Nearly 3 million people

lost their lives in the last two decades [16]. Considering this, modern technology can augment the existing disaster infrastructure, aiding in SAR operations.

Researchers have demonstrated that multi-UAV systems can be highly effective in providing support services in disaster environments. This is due to their efficiency, ability to operate simultaneously, flexibility, and fault tolerance. Disaster environments, such as SAR missions, are highly dynamic and unpredictable. New tasks may arise, UAVs may malfunction, and new UAVs may need to be added to the system. In [13], the authors presented a task allocation algorithm that can handle the arrival of new tasks during task execution. However, this algorithm can be further improved by incorporating more dynamic parameters, which is the motivation behind this paper.

B. Contribution of the article

This paper discusses the allocation of tasks to a multi-UAV system in a constrained and dynamic scenario such as Search and Rescue (SAR). In such a scenario, multiple parameters can change during the execution of tasks. The main objective is to handle the variation in such parameters and reassign tasks that have already been assigned to a UAV if it is unable to perform the task due to certain constraints. The paper highlights the following contributions:

1) **Fault Detection**: In order to enhance the effectiveness of CDPI algorithms, it is important to develop a method for detecting faulty UAVs during the task execution phase. A collaborative strategy is used to overcome this limitation. Each UAV that is part of the system shares locally computed time-cost information with other UAVs during the communication and conflict resolution phase. This sharing of information creates a robust mechanism that can identify a malfunctioning UAV based on the time-cost vector it shares with other UAVs.
2) **Enhanced Dynamic Activity Detection**: The CDPI algorithm has undergone significant improvements in detecting dynamic activity function. This enhancement allows the algorithm to efficiently reassign tasks that were initially allocated to a malfunctioning UAV. This ensures that the overall system remains functional.
3) **Adaptability in ECDPI Algorithm:** The ECDPI algorithm introduces a novel mechanism to accommodate variations in task duration and deadline. If a UAV encounters difficulties while completing a task due to these variations, the detect dynamic activity function is crucial in swiftly reassigning the task, ensuring adaptability in the presence of dynamic operational conditions.
4) **Scalability Feature:** A new feature has been added to the detect dynamic activity function to deal with changes in the number of tasks and UAVs in the multi-UAV system. This improvement ensures that the algorithm is scalable and effective, enabling it to handle changing operational scenarios with ease.

C. Organization of the Article

This paper is organized as follows: In section II, the dynamic task allocation problem and existing approaches are discussed with an emphasis on the types of parameters considered and their intended applications. In section III, a detailed mathematical description is presented. In section, IV, the existing compromised dynamic performance impact (CDPI) algorithm is described along with its limitations, and an enhanced CDPI task allocation algorithm is proposed. In section V, extensive simulations are presented against different scenarios, along with related discussions. Concluding remarks and future directions are presented in section VI.

II. Related Work

Dynamic task allocation is a crucial aspect of multi-UAV systems that operate in dynamic environments. In these situations, UAVs must adapt their behavior to environmental changes to enhance the overall performance of the system. The work in [17] explores multitasking to solve multiple optimization problems simultaneously and discusses that exchanging knowledge among tasks optimizes existing synergies and leads to optimal solutions.

The authors of [18] addressed the problem of multi-robot task allocation (MRTA) for the SAR domain. The objective was to enable a team of robots to collaboratively search for targets within a designated area and retrieve them back to their home base. The authors utilized two methods to allocate tasks to a group of robots: prediction-based and auction-inspired task allocation. To distribute tasks to each robot, they used an auction-based approach that relied on a winner-determination method.

In [19], the authors used a hedonic game theory-based autonomous decision-making framework to solve task allocation problems for a swarm of multiple agents. The proposed framework employs a self-organizing approach to enable agents to make decisions based on their individual preferences. In [12], the authors propose a PI algorithm to allocate tasks to a multi-UAV system in search and rescue scenarios under time constraints.

The work in [20] identified two shortcomings in the PI algorithm, namely the static structure and the possibility of getting trapped in local minima. To address these issues, the authors have introduced an online task rescheduling mechanism that involves a task reassignment methodology during the task execution phase. This mechanism allocates new tasks to UAVs, ensuring a more efficient operation. Additionally, to avoid getting trapped in local minima, the authors introduced a soft-max action selection method. This method enhances the exploratory properties of the PI algorithm, making it more adept at dealing with dynamic events like the addition or removal of tasks, updated task locations, and the removal or addition of UAVs. The inclusion of the soft-max action algorithm has made the PI algorithm more flexible and better equipped to handle real-world scenarios.

The authors of [21] developed a consensus-based bundle algorithm that allows multiple robots to work together on a task. This algorithm has been further improved by [22] to address critical time constraints and prevent issues with local minima. The proposed method uses dynamic grouping to allocate tasks based on changing robot states, task information, and network status.

The work in [23] presented a solution to the problem of online dynamic MRTA problem. The authors proposed a method called the DYMO-Auction algorithm, which takes into account several parameters, including task quality requirements, travel distance, and load balancing. The proposed algorithm was tested in a scenario where tasks appear dynamically along with their quality requirements. The algorithm assigns tasks based on a utility function that considers four parameters, namely, distance, energy, cost, and task type. These parameters are combined using a weighted sum model to calculate the cost function.

The authors in [24] proposed a method for efficient dynamic task allocation for a swarm of robots working on large-scale tasks. They used the concept of optimal mass transport theory for this purpose. The authors divided the tasks into smaller groups, known as regional tasks, to simplify the allocation process. The optimal mass transport theory approach is beneficial in unknown environments because it dynamically clusters the tasks into regional tasks, reducing the computation load.

In a scenario where a team of robots is exploring and destroying targets, MRTA can present a challenge. The robots have some knowledge of suspicious locations, which they explore before being assigned tasks, but they do not have an exact distribution of targets [25].

In dynamic and time-constrained scenarios, it may become necessary to reassign some of the tasks that have already been assigned if new tasks arrive during the execution of existing tasks. By doing so, empty time slots are created in the task lists of UAVs, where new tasks can be accommodated. The authors of [15] introduced the concept of task reassignment in the PI algorithm for assigning dynamically arrived tasks for distributed multi-UAV systems. They identified five types of dynamic events that can occur, including deletion of a task, addition of new tasks, updates to task deadlines, changes to task locations, and changes in task durations.

The authors in [26] proposed a dynamic algorithm for task allocation for unmanned underwater vehicle swarms. They achieved this by extending the CBBA algorithm and considering multiple parameters such as time, path, and UUV tour as marginal utility functions to optimize the baseline CBBA algorithm further. On the other hand, in [27], a stochastic conflict-based task allocation algorithm (SCoBA) was proposed to address the dynamic task allocation problem of a multi-agent system under task completion uncertainty and time window constraints. The objective of the algorithm is to minimize the number of unsuccessful tasks. The authors achieved this by decoupling sequential decision-making under multi-agent coordination and task uncertainty for multi-agent task allocation.

III. Mathematical Description and PI Algorithm Architecture

Consider a scenario with M survivors (tasks), denoted by $\mathbf{t} : [t_1, t_2, ..., t_M]$, and N UAVs available for rescue operations, represented by $\mathbf{u} : [u_1, u_2, ..., u_N]$, as shown in Figure 1. This is a SAR operation, where each task has a specific deadline. The UAVs are required to assist the survivors before their respective deadlines. If a task is not completed before its deadline, it will be considered a failure. The deadlines for all M tasks are represented by $\mathbf{d} : [d_1, d_2, ..., d_M]$.

A UAV can only perform one task at a time, and a task cannot be assigned to more than one UAV. As soon as a new task arrives, it needs to be assigned immediately. This method of task allocation is known as the single-task, single-robot instantaneous-assignment (ST-SR-IA) method [28]. In the case of using different types of UAVs, a compatibility matrix H is defined to determine the compatibility between a task and a UAV. The compatibility matrix has dimensions of $N \times M$.

The compatibility matrix has entries of either 1 or 0, depending on whether a particular task can be performed by a UAV or not. In the case of a homogeneous UAV system, all entries are 1. Since UAVs have limited carrying capacity and battery life, there is a cap on the maximum number of tasks a UAV can perform. Therefore, each UAV is assigned a specific number of tasks, subject to the battery limit. These constraints ensure optimal performance and efficiency of the UAV system.

In a SAR scenario, each task may have a different level of priority. Therefore, it is assumed that every task has a priority. The tasks allocated to a UAV are stored in a task list in a specific order, and they are executed in the same sequence. The task lists of N UAVs are represented as $\mathbf{s} : [s_1, s_2, ..., s_N]$. The tasks assigned to a UAV are performed in a way that ensures the deadline requirement is met at the lowest possible task cost. The cost $C_{i,q}$ of performing task t_q is calculated by adding the time that a UAV u_i takes to reach the task location and the duration T_{dur} needed to complete the task.

The main server broadcasts all information related to tasks, including the deadline, location, and priority, to all UAVs. The goal of this study is to assign several tasks to a multi-UAV system in such a way that the sum of priorities of the assigned tasks is maximized while adhering to the constraints mentioned in Eq. (1) [13].

$$
\begin{aligned}
\mathbf{J} &= \max \sum_{i=1}^{N} \sum_{k=1}^{|s_i|} p_{i,k}(s_i) \\
&subject\ \ to \\
&\mathsf{C}_1 :\ |s_i| \leq L \\
&\mathsf{C}_2 :\ c_{i,k}(s_i) \leq d_{si,k} \\
&\mathsf{C}_3 :\ h_{i,j} = \begin{cases} 1 & if\ u_i\ is\ compatible\ with\ t_j \\ 0 & o.w \end{cases} \\
&\mathsf{C}_4 :\ \sum_{k=1}^{|s_i|} c_{i,k}(s_i) \leq B_i \\
&\mathsf{C}_5 :\ s_i \cap s_j = \emptyset \quad \forall \quad i \neq j
\end{aligned}
\tag{1}
$$

Where constraint C_1 represents each UAV that can perform at most L number of tasks, constraint C_2 represents each assigned task to be performed before its deadline has passed. For constraint C_3, each UAV can only perform compatible tasks in the case of a heterogeneous multi-UAV system. The constraint C_4 describes that each UAV can perform a number

Fig. 1: Proposed Multi-UAV Task Allocation Architecture.

of tasks within its battery limit, whereas the constraint C_5 describes that a task t_q cannot be assigned to more than one UAV.

A. Compromised Dynamic Performance Impact Algorithm

In a dynamic environment such as SAR, various parameters related to the task may change during its execution. For instance, the arrival of new tasks, changes in task duration or task deadline, and other factors may affect task allocation. Therefore, it is crucial to allocate tasks while considering such changing dynamics. Additionally, the task allocation algorithm must also accommodate tasks assigned to a faulty UAV or assign tasks when a new UAV is added to the system. Similarly, the algorithm must handle new tasks that arrive or tasks that are deleted during the execution of the tasks.

In [13], the CDPI algorithm is introduced as a method of allocating tasks during the execution phase. It is an extension of the compromised PI (CPI) algorithm, which is a static task allocation method. The CPI algorithm allocates tasks based on the VIKOR method, considering multiple constraints and attributes of both UAVs and tasks. The PI algorithm, which is also a static task allocation algorithm, is the baseline for both the CPI and CDPI algorithms. The PI algorithm allocates tasks based on only one parameter, which is the time cost. Each UAV runs the CDPI task allocation algorithm as its main routine, and Algorithm 1 outlines the details of this algorithm.

The main algorithm consists of two parts. The first part is a PI algorithm $(lines\ 4-10)$, which is a static task allocation algorithm. It assigns tasks before task execution. After UAVs commence task execution, the second part of the CDPI algorithm $(lines\ 11-22)$ allocates dynamically arrived tasks. The CDPI algorithm terminates when all assigned tasks are executed.

Algorithm 1: CDPI Task Allocation-Main Routine

```
1  Define ⇒ u, t, ℘_i, β_i, τ
2  Initialize UAVs: ⇒ Type, Location, Battery Limit, Task
     Limit
3  Initialize UAVs Tasks: ⇒ Type, Location, Deadline,
     Priority, T_iv
4  Iteration = 0;
5  while CPI-Consensus is false do
6  |   Compromised Task Inclusion Phase
7  |   Communication & Conflict Resolution Phase
8  |   Iteration ++
9  |   CPI − Consensus⇐ CheckConsensus
10 end
11 T_iv ⇐ Mask Assigned Tasks
12 while All Tasks not Executed do
13 |   Detect Dynamic Activity ⇒ Algorithm 4
14 |   while Unassigned Tasks do
15 |   |   IPI ⇒ Algorithm-7 of [13]
16 |   |   Communication & Conflict Resolution Phase
17 |   |   T_iv ⇐ Mask Assigned Tasks
18 |   |   Iteration ++
19 |   |   if Improvement = ∅ then
20 |   |   |   Break
21 |   |   end
22 |   end
23 end
```

The CDPI algorithm builds a task list on each UAV by iteratively performing the task inclusion, communication, and conflict resolution phases to allocate tasks received before task allocation. The task assignment process runs on all UAVs until there are no tasks left to assign or the solution cannot be improved. The dynamic portion of the CDPI algorithm allocates tasks arrived during task execution. This process is executed until all assigned tasks have been performed. In the

CDPI algorithm, the detect dynamic activity function identifies any dynamically arrived task, given in Algorithm 4 ($lines\ 1-14$).

In the baseline PI and CDPI algorithms, every UAV keeps track of two global vectors to monitor its assigned tasks. These vectors are called the removal PI (RPI) vector and the UAV-ID vector. To better comprehend the CDPI algorithm, it is essential to understand the concept of UAV-ID vector, task list, and the baseline PI task costing mechanism.

- **UAV ID-Vector**: When a UAV computes the cost of a task t_q less than its globally achieved task cost, it includes t_q in its task list s_i and updates its UAV-ID vector against task t_q accordingly. The length of the UAV-ID vector is equal to the total number of tasks.
- **Task List**: During the task inclusion phase, a number of tasks are assigned to each UAV, and these tasks are stored in the task list s_i of each UAV, where i is the i^{th} UAV u_i. The assigned tasks are ordered in s_i in such a way that the overall cost of performing all the tasks by a UAV is minimized. The task inclusion process is described in the following example.
  Consider that there are three tasks t_3, t_6, and t_1 in the task list s1 of a UAV u_1 in the prescribed order, as shown in Figure 2. Assume that task t_4 can be included at all positions marked with vertical arrows in the task list and u_1 performs all tasks with the described time cost, which is the time to reach the task location from the previous location. Here, time cost is taken as the sum of time to reach a task position and task duration T_{dur} to perform that task.
- **Baseline PI Costing Mechanism**: The task costing mechanism of the CDPI algorithm is the same as that of the PI algorithm. In Figure 2, it is shown that three tasks t_3, t_6, t_1 are included in the task list of u_1 along with their travel time cost from one point to another point. Task t_4 is required to be included in a task list and assume that it can be placed at all positions in the task list, conforming to all the constraints. It is clear from Figure 2, that with the addition of a new task, the overall time cost to perform all tasks is increased. Assuming that T_{dur} is equal to 100 sec, the cost to perform task t_3 and t_6 is 170 and 330 seconds [14].
  The overall time cost to perform all the tasks in the task list after insertion of t_4 at 1^{st} position is given as:-

$$c_1(s_1 \oplus 1, t_4) = c_{1,4}(s_1) + c_{1,3}(s_1) + c_{1,6}(s_1) + c_{1,1}(s_1) \quad (2)$$

  where $c_1(s_1 \oplus 1, t_4)$is the cost when task t_4 is added at 1^{st} position in the task list of u_1. Since task t_4 can be included at all positions in the task list, the addition of a task increases the overall time cost to perform tasks. So, the decision of task position is made based on the minimum increase in total cost, which is computed as:

$$l_{\min} = \min[c_1(s_1 \oplus l, t_4) - c_1(s_1)], \quad l = 1 : |s_1|+1 \quad (3)$$

  where l is the task position of task t_4 inserted in the task list and $|s_1|$ is the number of tasks in the task list. It is pertinent to mention that only those positions of l are considered where all constraints are conformed.
- **Inclusion Performance Impact (IPI)**: CDPI algorithm uses the costing mechanism of PI algorithm where IPI of task t_q in task list s_i is defined as the overall increase in the time cost of tasks with the addition of $t_q$at position l. The IPI of task $t_q$for inclusion in the task list s_i is mathematically computed as [13]:

$$v^*_{q,l}(s_i, t_q) = \sum_{x=l}^{|s_i|+1} c_{i,x}(s_i \oplus l, t_q) - \sum_{x=l}^{|s_i|} c_{i,x}(s_i) \quad (4)$$

  where $s_i \oplus l, t_q$ gives inclusion of t_q in s_i of UAV u_i at position l. The minimum IPI value of the task t_q is

$$v^o_q(s_i, t_q) = \min_{l=1}^{|s_i|+1} v^*_{q,l}(s_i, t_q) \quad (5)$$

  By using Eq. (4) and (5), IPI values of all tasks for inclusion in the task list are computed and stored in vector form as $\wp^o_i : v^o_1, v^o_2, ..., v_(q,)^o, .., v^o_m$, where v^o_q is the IPI value of task t_q in s_i of UAV u_i.
- **Removal Performance Impact (RPI)** : Once the task list of a UAV is full, or no further tasks can be included in the task list, each UAV computes the RPI value of tasks included in its task lists for sharing with other UAVs and is given as [13]:

$$v^{\ominus}_q(s_i, t_q) = \sum_{x=b}^{|s_i|} c_{i,x}(s_i) - \sum_{x=b+1}^{|s_i|} c_{i,x}(s_i \ominus t_q) \quad (6)$$

B. Compromised Task Inclusion Phase

In the static part of the main CDPI algorithm, the compromised task inclusion phase is first executed for available tasks. The PI algorithm selects a task for inclusion in s_i based on minimum task cost, and similarly, the TRMaxAlloc algorithm [29] selects for inclusion in the task list based on time cost. In compromised task inclusion, a task is selected for inclusion in the task list at position l task based on task and UAV attributes and constraints. VIKOR method is used, which gives a rating of tasks based on task and UAV attributes and constraints, i.e., task priority, task urgency, task cost, and UAV remaining battery time. The compromised PI task inclusion phase is given as Algorithm 2.

In order to include a task in the task list using Algorithm 2, the IPI vector of tasks is first computed using Algorithm 2 of [13], a task is included at all positions in the task list, and if all constraints are satisfied, then corresponding IPI values are computed. The task with a minimum increase in computed cost at any position is included in the IPI vector. Similarly, IPI values are computed for all tasks. The number of elements in G can be up to M, i.e., total number of tasks:

$$G = (\wp_{i,q} - \wp^o_{i,q}) \geq 0 \quad (7)$$

In order to select a task from G Eq. (7), the VIKOR method is used, which is used here to rank the tasks based on considered constraints and attributes. A task with a minimum VIKOR value is selected for inclusion in the task list. After the inclusion of a task in the task list, the UAV-ID vector and time

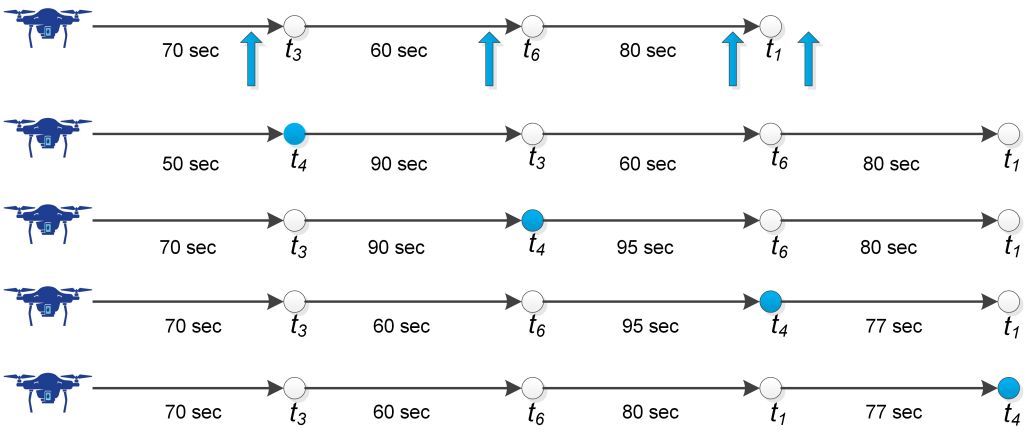

Fig. 2: Task Location in S_1 of UAV u_1.

Algorithm 2: CPI Task Inclusion Phase

Input s_i, L, $\wp_i$, β_i, C_i
Output β_i, $\wp_i$, s_i, C_i
while *No. of Tasks in* $s_i \leq L$ **do**
  Compute $\Rightarrow$ IPI using Algorithm-2 of [13]
  Compute $\Rightarrow$ G
  if $G \neq \emptyset$ **then**
    Select $Task$ using Vikor Method
    Insert t_q at Position l
    Update β_i & C_i
  else
    Exit
  end
end
Update $\leftarrow$ $\wp_i$

cost vector are updated, where the time cost vector stores the cost values to perform a task. This process is repeated again to include more tasks until the task list is full or no new task can be included.

In the end, when either the task list is full or further tasks cannot be included, the RPI values of tasks corresponding to the tasks in the task list are updated. The communication and conflict resolution phase is executed once the capacity to include tasks in the task list of a UAV is either full or the solution cannot be improved.

C. MCDM VIKOR Method

In the VIKOR algorithm, the Multi-Criteria Decision-Making (MCDM) problem is defined as [13]:

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ b_{M1} & b_{M2} & b_{M3} & b_{M4} \end{pmatrix} \tag{8}$$

where, b_{ij} are the set of alternatives that qualify Eq. 7 and columns of Eq. (8) are $P = [P_1, P_2, P_3, P_4]$ and defined in [13]. The task is rated by the VIKOR method using the following steps as defined by authors in [30]:

1) In the first step of the VIKOR method, the best and worst value of each alternative/criterion is determined, i.e.,
   - For beneficial alternative/criteria
   $$b_j^* = \max_j b_{ij} \quad \text{and} \quad b_j^- = \min_j b_{ij} \tag{9}$$
   - For non-beneficial alternative/criteria
   $$b_j^* = \min_j b_{ij} \quad \text{and} \quad b_j^- = \max_j b_{ij} \tag{10}$$

   Beneficial and non-beneficial criteria can be described using the following example. Let's say there are two criteria for task selection: task priority and cost. The priority of a task is the beneficial criterion; therefore, a task with higher priority is selected, whereas, for the non-beneficial criterion, a task is selected with low task cost.
2) In the second step, utility measure value Y_i and regret measure value Z_i of each alternative is calculated as:
   $$Y_i = \sum_{j=1}^{n} W_j \left(\frac{b^* - b_{ij}}{b_j^* - b_j^-} \right) \tag{11}$$
   $$Z_i = \max_j \left[W_j \left(\frac{b^* - b_{ij}}{b_j^* - b_j^-} \right) \right] \tag{12}$$
   Where W_j is the weight of each described criterion. In the described case, with the suitable value of weights W_j, tasks can be allocated in such a way that the overall sum of priorities is maximized.
3) In step three, the VIKOR index is computed as follows:
   $$\partial_i = \nabla \left[\frac{(Y_i - Y^*)}{(Y^* - Y^-)} \right] - (1 - \nabla) \left[\frac{(Z_i - Z^*)}{(Z^* - Z^-)} \right] \tag{13}$$
   where ∇ is defined as:
   $$\partial_i = \begin{cases} \nabla > 0.5 & \text{Majority rule} \\ \nabla \approx 0.5 & \text{Consensus rule} \\ \nabla < 0.5 & \text{Veto rule} \end{cases} \tag{14}$$
4) In the fourth step, the alternative is selected with the smallest value of ∂. After the fourth step, the VIKOR

Algorithm 3: Communication & Conflict Resolution Phase

Input & Output s_i, C_i, $\wp_i$, β_i
Send $\Rightarrow$ **UAV** $\rightarrow$ $\wp_i$, β_i, C_i
Recieve $\Leftarrow$ **UAV** $\leftarrow$ $\wp_i$, β_i, C_i
Find Task to Remove
while *all tasks are removed* **do**
    Remove Task from s_i
    Update $\wp_i$, C_i, β_i
end

method ranks tasks in a compromised way based on the considered attributes and constraints. Using this method, a compromised task is selected from a computed IPI vector that conforms to all constraints and conditions. As the VIKOR method is a ranking algorithm, more parameters/constraints can be included without any modification in the CDPI algorithm.

D. Communication & Conflict Resolution Phase

When all tasks have been executed, or no further task can be allocated by a UAV, the communication and conflict resolution phase is executed then. In this phase, each UAV in a multi-robotic system broadcasts its RPI and $UAV-ID$ vectors to all other UAVs. Each UAV updates its RPI and $UAV-ID$ vectors after receiving these vectors from connected UAVs against the minimum RPI value achieved so far. A task is removed from the task list of a UAV when the computed RPI value of that task is higher than its received RPI value. A UAV u_i updates its cost vector C_i, once a task is removed from its task list. The communication and conflict resolution phase is given in Algorithm 3.

Each UAV continues its task inclusion, communication, and conflict removal processes until either all tasks have been allocated or no further improvement can be made. In the static part of the task allocation algorithm, when consensus has been achieved, UAVs start executing the allocated tasks. The time cost vector is shared with other UAVs in the proposed ECDPI algorithm. When UAVs start executing tasks, the dynamic part of the CDPI algorithm runs in parallel until all tasks have been executed. When a task is executed, the respective UAV sets its RPI value to zero and communicates to other UAVs. All UAVs set the value of an executed task in task inhibit vector T_{iv} as one, so that these tasks do not become part of task reassignment, where task reassignment is necessary to accommodate newly arrived tasks in the task list. Therefore, task inhibit vector T_{iv} eliminates tasks from the task assignment process.

A dynamic activity is detected using Algorithm 6 as proposed in [13] and keeps executing until all tasks have been executed. Detect dynamic activity algorithm's primary role is to detect new tasks and inform other UAVs about the completed tasks. When a new task is detected, each UAV sets the RPI value of un-executed tasks to a maximum value. During the allocation of newly arrived tasks, it may be necessary to reassign non-executed tasks to other UAVs, which creates a feasible time slot for the new tasks.

By setting the RPI value of un-executed tasks at maximum, the task assignment process is re-initiated. In the CDPI algorithm, it is assumed that a task t_q is considered executed once it starts moving towards that task by assuming UAVs never fail. By continuously communicating RPI values to other UAVs, each UAV keeps track of all executed tasks in the system.

In detecting dynamic activity function, RPI values of un-executed tasks are set to maximum, and after that, the task inclusion phase is executed. In the dynamic part of the CDPI algorithm, the IPI vector is computed using Algorithm 7 of [13].

IV. Enhanced Compromised Dynamic Performance Impact Algorithm

The CDPI algorithm focuses on one specific dynamic parameter the allocation of newly arrived tasks during task execution. However, in SAR scenarios, there may be other dynamic parameters to consider, such as variations in task duration, task deadlines, and potential faults in UAVs. Therefore, a task allocation algorithm must take into account these variables during the task execution phase.

This study enhances the CDPI algorithm by considering various dynamic events that occur during the task execution phase, such as variations in task duration and deadlines, faults in UAVs, deletion of tasks, and inclusion of new UAVs into the system.

A. Occurrence of fault in UAVs

The CDPI algorithm assumes that UAVs will not encounter any failure and will accomplish their designated tasks. However, in reality, UAVs can fail during task execution, and there should be a mechanism in place that enables the assigned tasks to be reassigned to other UAVs available in the system.

In CDPI and baseline PI algorithms, UAVs only share the RPI value of each task along with the assigned UAV, and each UAV performs its allocated tasks as per the sequence in the task list. UAVs do not share the sequence of allocated tasks with each other, so a UAV cannot know about the current state of task performance for other UAVs. However, if UAVs can share the cost vector of their assigned tasks, then each UAV can clearly find the current task being performed by a UAV along with the expected finish time. The expected finish time of a task is considered because of the assumed variation in the task duration due to the dynamic environment.

In the CDPI algorithm, after completion of a task, each UAV sets its RPI value to zero and sends this updated RPI vector to all other UAVs. Each UAV then sets the corresponding value in the task inhibit vector so as to avoid considering this task in case of any dynamic activity. In the case of mesh topology, where all UAVs can directly communicate with each other, any completed task or dynamic event information is instantaneously available for each UAV in the multi-robotic system.

On the other hand, when UAVs can only communicate with neighboring UAVs, some iterations are required for the communication of information to far-end UAVs. If each UAV shares the time cost of its assigned tasks, it is exactly known

to all UAVs when a task is going to be completed. Further, in the CDPI algorithm, each UAV shares its task completion status with all other UAVs.

Let us assume that T_{cd} is the time required in a multi-UAV system for communication of dynamic information, i.e., task completion, to all UAVs. A UAV will be declared faulty if its assigned task completion status is not received by all UAVs till T_{di} time is given as:

$$T_{di} = c_{i,q}(s_i) + T_{cd} \tag{15}$$

Where $c_{i,q}(s_i)$ is the completion time of task t_q assigned to UAV u_i and T_{di} is the time at which dynamic information is available to all UAVs. With the knowledge of task completion time for each task assigned to any UAV, along with the sharing of task completion messages, it becomes possible to determine any faulty UAV. Detect dynamic activity algorithm is enhanced to determine any faulty UAV in the system as Algorithm 4 ($line\ 15-19$). When a UAV is declared faulty, all UAVs will set RPI values of tasks assigned to faulty UAV to a maximum, along with corresponding UAV-ID vector values to zeros.

Figure 3 shows four UAVs along with their assigned tasks. It is clear from the figure that the time cost to perform each task is different, and at any instant, say at the blue vertical arrow, UAV u_2 has just completed the task t_2 and is heading towards t_5. Whereas UAV u_3 is moving towards its first task location, i.e., t_8.

In Figure 3, it is shown that UAV u_1 failed while it was heading towards its first task location, as marked by a red vertical arrow. But all other UAVs in the system will conclude UAV failure when they do not receive a task completion message until T_{di} time, as marked by a blue vertical arrow. Assume that it requires Treas time for reassignment of tasks assigned to faulty UAV, shown as a green vertical arrow in Figure 3. It is clear in Figure 3 that UAV u_4 was approaching towards t_{10} when UAV u_1 was declared faulty. Whereas, when task reassignment and consensus phase finished, it was heading towards t_{13}. Similarly, at time T_f, UAV u_3 is about to reach the location of task t_8. The final time at which all dynamic changes have been incorporated into the system is:

$$T_f = T_{reas} + T_{di} \tag{16}$$

Each UAV will set RPI values of un-executed tasks assigned to a faulty UAV to maximum and corresponding UAV-ID values to zero. In order to maximize the assignment of these tasks, each UAV u_i will also set RPI values of its un-executed tasks in its task list s_i starting after T_f time to a maximum value. For tasks currently executing or which are going to start before T_f time are not part of this reassignment process, and their values in T_{IV} will be set 1. After setting the RPI values of tasks at maximum, task inclusion and communication and conflict resolution phases are executed. Only those tasks that become part of the task reassignment process whose RPI values are at maximum.

Algorithm 4: ECPI-Detect Dynamic Activity

```
1  χ* = 0
2  while χ* = 0 do
3      if New Task then
4          Set ℘_i == Max (Task starts after T_f)
5          χ* = 1
6      end
7      if Task Executed then
8          Set ℘_i (Task) = 0
9          T_iv (Executed Task) =1
10         Send/Received Executed Task
11     else if Executed Task then
12         Set ℘_i (Task) = 0
13         T_iv (Executed Task) =1
14         Send/Received Executed Task
15     else if UAV Failed then
16         Set ℘_i == Max (Task Assigned to Failed UAV)
17         β_i =0 (Tasks Assigned to Failed UAV)
18         Set ℘_i == Max (Task Starts after T_f)
19         χ* = 1
20     else if Variation in Task Duration then
21         if UAV can't Execute Task
22         Set ℘_i == Max (Task with Increase Duration)
23         β_i =0 (Task with Increase Duration)
24         Set ℘_i == Max (Task Starts after T_f)
25         χ* = 1
26     else if Variation in Task Deadline then
27         if UAV can't Execute Task
28         Set ℘_i == Max (Task with Earlier Deadline)
29         β_i =0 (Task with Earlier Deadline)
30         Set ℘_i == Max (Task Starts after T_f)
31         χ* = 1
32     else if Deleted Task then
33         if any un-assigned Task
34         Set ℘_i == Max (Task Starts after T_f)
35         χ* = 1
36     else if New UAV then
37         if any un-assigned Task
38         Set ℘_i == Max (Task Starts after T_f)
39         χ* = 1
40     Send/Receive ℘_i
41     Send/Receive β_i
42 end
```

B. Variation in the Task Duration

Due to variations in the environment or unforeseen reasons, it is possible that a task requires more time to be performed than expected. In variations in the task duration, either a task can take a longer or less time to perform a task against its estimated task duration. In the considered scenario, tasks are subject to their deadlines and are to be performed before their deadlines expire. UAVs include multiple tasks in their task list before execution of tasks. The variation in task duration is communicated to UAVs via the central server.

In case a task t_c requires less time than the estimated task duration, it will not affect the tasks (if any) later in the task list. In case it takes longer than its task duration, then there can be multiple situations given as:

- The t_c is only one task: In this case, if the extension in task duration is within the UAV's battery limit, the UAV will perform that task. Otherwise, it will set the RPI value of the task to the maximum so that other UAVs can include that task in their task list.
- The t_c is the last task: In this case, if the extension in task duration is within the UAV's battery limit, the UAV will perform that task. Otherwise, it will set the RPI value of

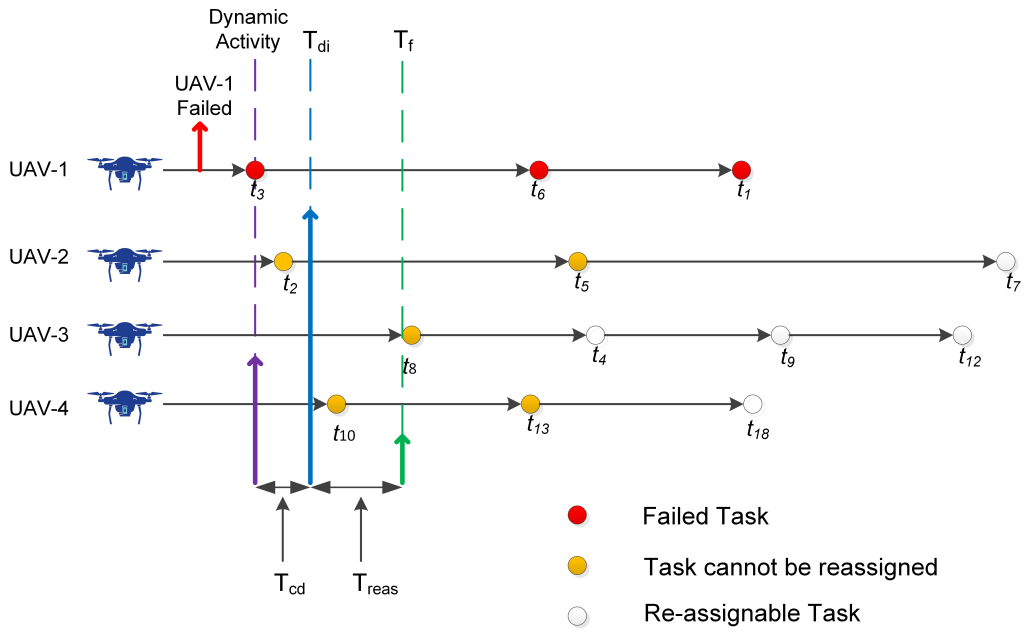

Fig. 3: Detection of Faulty UAV.

the task to the maximum so that other UAVs can include that task in their task list.

- There is at least one task after the t_c: In this case, if the extension in the task duration is not affecting the deadline of tasks after t_c or the battery limit of UAV is not violated, the respective UAV will perform all of its assigned tasks. On the other hand, if this extension in deadline results in affecting the deadline of other tasks or the UAV battery limit, then the RPI value of t_c will be set to maximum for reassignment and communicated to all other UAVs.

In case a UAV is unable to perform a task, it will immediately set its RPI value to the maximum and communicate to other UAVs as given in Algorithm 4 ($lines$ $20-25$). Assuming there is a negligible time between receiving task duration information from the server and setting RPI value, then it will take a maximum Tcd time to communicate this information to all other UAVs in the system. In order to maximize the chance to reallocate task t_c, all UAVs will set RPI values of their un-executed tasks starting after T_d time to maximum, where T_d is given as:

$$T_d = T_{reas} + T_{cd} + T_{ss} \tag{17}$$

In Eq. (17), T_{ss} is the time when the central server broadcasts dynamic information, and T_d is the time when dynamic information has been incorporated in the system. In case task t_c cannot be reassigned to any UAV, the respective UAV will perform other tasks assigned to it, excluding t_c from the execution process.

C. Variation in Task Deadline

In this scenario, every task has a specific deadline that must be met in order for the task to be considered completed. If a task is not completed by its deadline, it is considered a failed task. The central server sends updates to all UAVs regarding any changes to the task deadlines. If a task's deadline changes while it is being executed, the execution process continues without any changes.

If the deadline for a task is earlier than originally set, but the assigned UAV is still able to perform the task within its limit, then the task will proceed as planned. However, if the deadline is moved closer and the assigned UAV is unable to complete the task within the new deadline, then the task will need to be reassigned to a different UAV. In this case, the original UAV will set its RPI value to the maximum to allow for reassignment. Additionally, all UAVs will set the RPI values of their tasks starting after time T_d to the maximum value, increasing the likelihood of reassignment for tasks with closer deadlines, as outlined in Algorithm 4 ($lines$ $26-31$).

D. Deletion of Tasks

The CDPI algorithm has the ability to assign new tasks to UAVs even after the task execution process has started. However, if a task is deleted from the system before being assigned to any UAV, the task execution process will continue as planned without any changes. On the other hand, if an assigned task is deleted from the system and there are still unassigned tasks available, all UAVs will set their RPI values to the maximum for the purpose of assigning unallocated tasks. This is described in detail in Algorithm 4 ($lines$ $32-35$). Additionally, all dynamic information is broadcast to UAVs by the central server.

E. Inclusion of New UAVs

If a new UAV is added to the system while there are still unassigned tasks, all UAVs will adjust the RPI values of their assigned tasks that come after a certain time T_d to the maximum value. This will initiate the task reassignment process

and increase the likelihood of more tasks being assigned. This process is described in Algorithm 4 ($lines\ 36-39$).

V. Simulation Results

In this section, we will discuss the performance of the ECDPI algorithm proposed for the SAR scenario against dynamic events such as task deadline variation, occurrence of faults, arrival of new UAVs, and deletion of tasks. The proposed ECDPI algorithm is implemented in MATLAB software, utilizing the CPU specifications of Intel core i5 - 8265U running at 1.8 GHz. The task execution phase involves the execution of assigned tasks by each UAV in parallel with the handling of dynamic events. The SAR scenario considered in this context is described in [13], [29].

The survivors of a disaster incident are located randomly within a defined area of $10km \times 10km \times 1km$. These survivors are in need of food or medical supplies. To address this, unmanned aerial vehicles (UAVs) will be used to deliver food and medical supplies to the survivors. The UAVs have a constant speed of 30 m/s for food delivery and 50 m/s for medical supplies. Each survivor requires 100 seconds to be served by a UAV upon reaching the location. Each UAV can serve a maximum of 8 survivors within its battery limit. The battery limit of each UAV is randomly set between 1500-2000 seconds.

Each task has its deadline, and the deadline for each task is randomly defined between 0 and 2000 seconds. Additionally, each task is assigned a priority value between 0 and 1. At the start of the simulation, UAVs are assumed to be located on the ground, specifically at a position of $10km \times 10km \times 0km$.

The communication topologies considered for the multi-robotic system are the same as those described in the baseline CDPI algorithm [13]. These include mesh, row, circular, and hybrid. During the static phase of the CDPI algorithm, UAVs communicate with each other using these communication topologies for task allocation. During the task execution phase, UAVs are distributed randomly based on their task list, so their communication is determined by the current configuration at that time. The VIKOR parameters used in all simulation scenarios are the same as those given in [13].

A. Simulation Scenario 1: Determination of Communication Time for Dynamic Information

In the CDPI algorithm, UAVs share their RPI and UAV-ID vectors to track the tasks they are assigned and the ones they have completed. However, in the proposed study, each UAV also shares cost vectors with all other UAVs to track any faulty UAVs during the task execution phase. In the CDPI algorithm, UAVs communicate with other UAVs only when a task is completed or when any other dynamic event occurs. If any dynamic event occurs, each UAV waits for T_{cd} time to ensure that all the required dynamic information is available to all UAVs in the system. After that, the task inclusion, communication and conflict resolution phases are executed.

It is necessary to determine the communication time between two distant UAVs (T_{cd}), since UAVs are in random configuration during task execution. To find T_{cd}, 10 UAVs were selected for 4 different topologies. A task was assigned to $UAV-10$ and the number of iterations required was recorded until all UAVs received the task completion update. The number of iterations required for each topology is presented in Table I.

TABLE I: No. of Iterations for Communication

Topology	**Mesh**	**Row**	**Circular**	**Hybrid**
No. of Iteration	1	9	5	5

The maximum number of iterations required to update the task completion information is for row topology, which is 9 iterations for 10 UAVs. It is because, in row topology, one UAV can only communicate with its immediate UAVs. If Δt is the time taken for one communication iteration between two UAVs, then a maximum communication time is taken against the dynamic configuration of the multi-UAV system, which is given as:

$$T_{cd} = \Delta t \times N - 1 \tag{18}$$

Let Δt be equal to $2s$ then T_{cd} is $18s$ for 10 UAVs. For all simulations in the paper, Δt is taken as 2 seconds, and a number of iterations of row topology are taken to calculate T_{cd}, respectively.

B. Simulation Scenario 2: Occurrence of Fault in UAV

In this scenario, a simulation is performed for the allocation of 16 tasks to 6 UAVs. Priority and deadline of each task are randomly set between $0-1$ and $0-2000s$, respectively, whereas the battery limit of each UAV is set between $1500-2000s$. UAVs and tasks are randomly placed in an urban disaster environment within the bounds as given in Table II.

The static part of the proposed ECDPI algorithm allocates 15 tasks to 6 UAVs in 11 iterations, and task t_3 is not assigned due to an earlier deadline, as shown in Figure 4. Table II gives tasks assigned to UAV u_6 and u_5 along with related data, i.e., task priority, deadline, time to reach the task location, etc.

The value of T_{cd} in this case is 10 seconds. Tasks t_{15} and t_{13} are assigned to UAV u_6, and at the start of the task execution phase, UAV u_6 is set to fail after the allocation of tasks during the start of the task execution phase. Since task t_{15} is not performed by UAV u_6 and all other UAVs get the knowledge after $123s$ (setting $c_{6,15}(s_6) = 113s$, $T_{cd} = 10s$) in Eq. 18 that UAV u_6 has failed In order to find which tasks are to be excluded from the task reassignment process, the parameter T_{reas} is the product of the number of iterations for task reassignment and the time for one iteration.

Due to the random task locations and different constraints, the number of iterations required for consensus on the reassignment of tasks during the task execution phase cannot be predicted. However, each UAV knows the number of iterations performed for consensus during static task allocation, and it is higher than consensus for task reassignment during task execution. Therefore, a number of iterations for static case task allocation is considered for the computation of Treas, whereas the time for one iteration is taken the same as that of ∇t, and hence $T_{reas} = 22s$. Thus, T_f is $145s$, and at this time, UAV u_5 is performing task t_{16} with t_{12} as re-assignable

TABLE II: Failure of UAV at the start of Task Execution

			Static Task Allocation		Dynamic Task Allocation		
UAV # 5	**14462 s**		**No. of Iterations**	11	**No. of Iterations**	6	
	UAV Battery Time	**Task**	T_{16}	T_{12}	T_{16}	T_{12}	T_{13}
		Arrival Time	71.60	226.1	53	210	414
		Priority	0.7252	1	0.7252	1	0.5106
		Deadline	748	851	748	851	574
UAV # 6	**9509 s**	**Task**	T_{15}	T_{13}	UAV Failed		
		Arrival Time	113	241			
	UAV Battery Time	**Priority**	0.3637	.5106			
		Deadline	175	574			

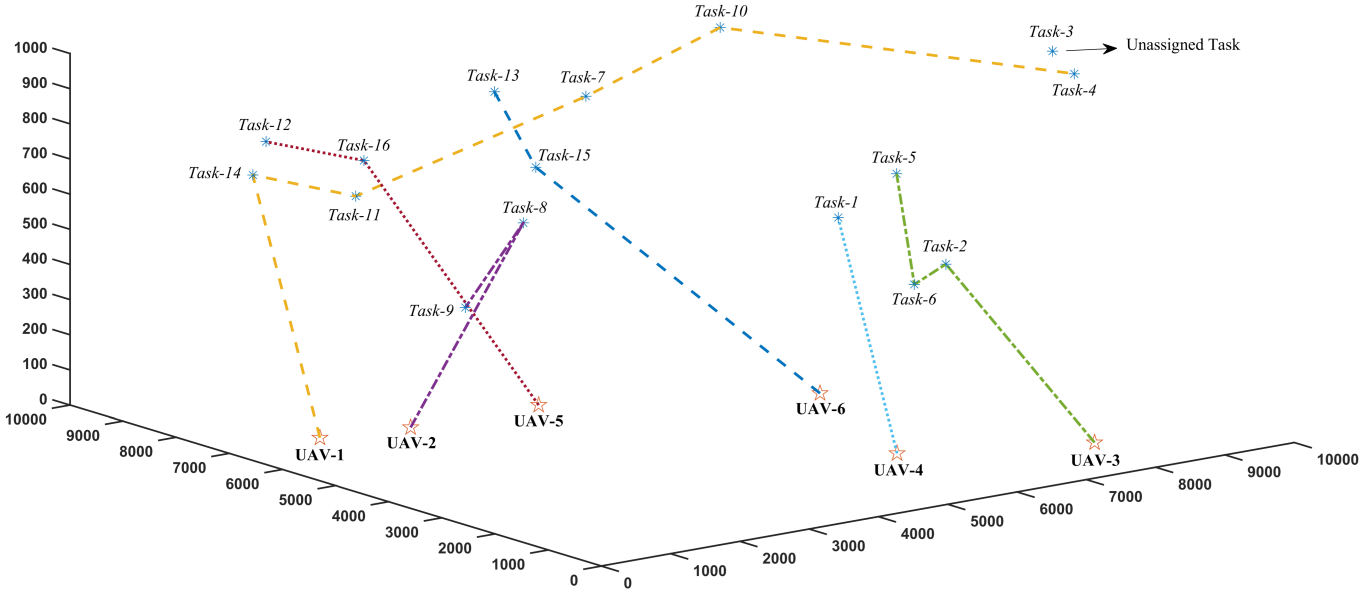

Fig. 4: Static Allocation for Simulation Scenario 1

tasks. Therefore, t_{13} is assigned to UAV u_5 at the position after task t_{12}, but t_{15} cannot be reallocated to any UAV due to an earlier deadline as shown in Figure 5.

C. Simulation Scenario 3: Variation in task duration

In this scenario, 16 tasks are considered for assignment to 6 UAVs, as shown in Figure 6. A total of 15 tasks are allocated to 6 UAVs using the static part of the ECDPI algorithm in 7 iterations. Therefore, the value of T_{cd} is $10s$ and T_{reas} is $14s$ respectively. Task t_{14} cannot be allocated to any UAV due to task deadline issues. At the start, the duration of all tasks is 100 seconds.

At the start of task execution, the task duration of task t_2 is increased to $130s$. Therefore, T_{ss} is zero, and T_d is equal to $24s$ respectively. With this task duration, UAV u_5 is not able to reach the location of task t_8 as given in Table III. Hence, t_8 is either required to be reassigned to any other UAV or be deleted from the task list of u_5. With T_d of $24s$, all tasks except the first in the task list of all UAVs are re-assignable. UAV u_6 is able to perform task t_2 in addition to its already allocated tasks within their deadlines. Therefore, t_2 is placed at the first location in the task list of UAV u_6, as shown in Figure 7.

D. Simulation Scenario 4: Variation in Deadline

The scenario of the task deadline is similar to the task duration. The only difference is with an extension in the duration of a task; the respective UAV may not be able to reach a task location that is placed later in the task list. In this case, the task with an extended duration is either reassigned to another UAV or removed from the task list. Similarly, when the deadline of a task is set early by the central server, the respective UAV may not be able to reach the task location before its new assigned deadline. In this case, either task is reassigned to another UAV or removed from the task list. Therefore, the simulation of the variation in deadline is not presented in the paper.

E. Simulation Scenario 5: Deletion of a Task

In this simulation scenario, 20 tasks are considered for assignment to 6 UAVs, as shown in Figure 8. Using the static part of the ECDPI algorithm, 19 tasks are allocated in 9 iterations, and one task, t_{16} cannot be allocated due to a deadline constraint as given in Table IV. In this case, T_{cd} is $10s$ and T_{reas} is equal to $18s$. It is clear from Figure 8 that tasks t_{16} and t_4 are very close to each other, but t_4 is assigned due to the early deadline and higher task priority.

At the start of task execution, task t_4 is deleted from the system. Therefore, T_{ss} is equal to zero and T_d is equal to $28s$

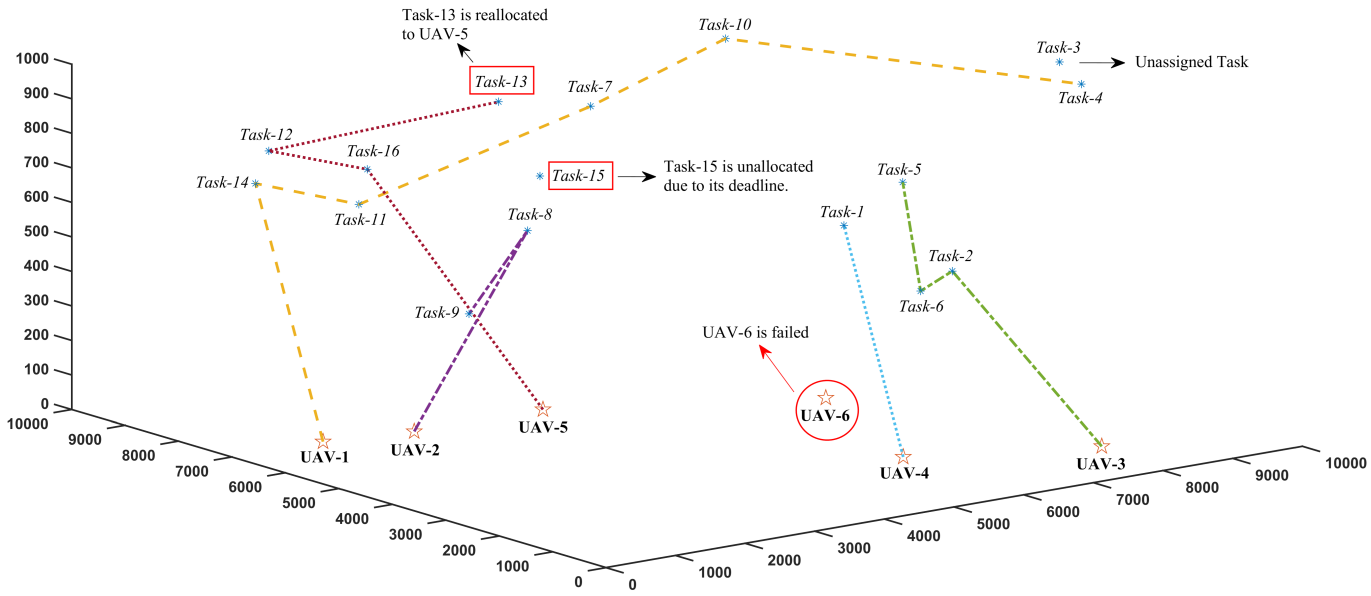

Fig. 5: UAV Malfunction during Task Execution

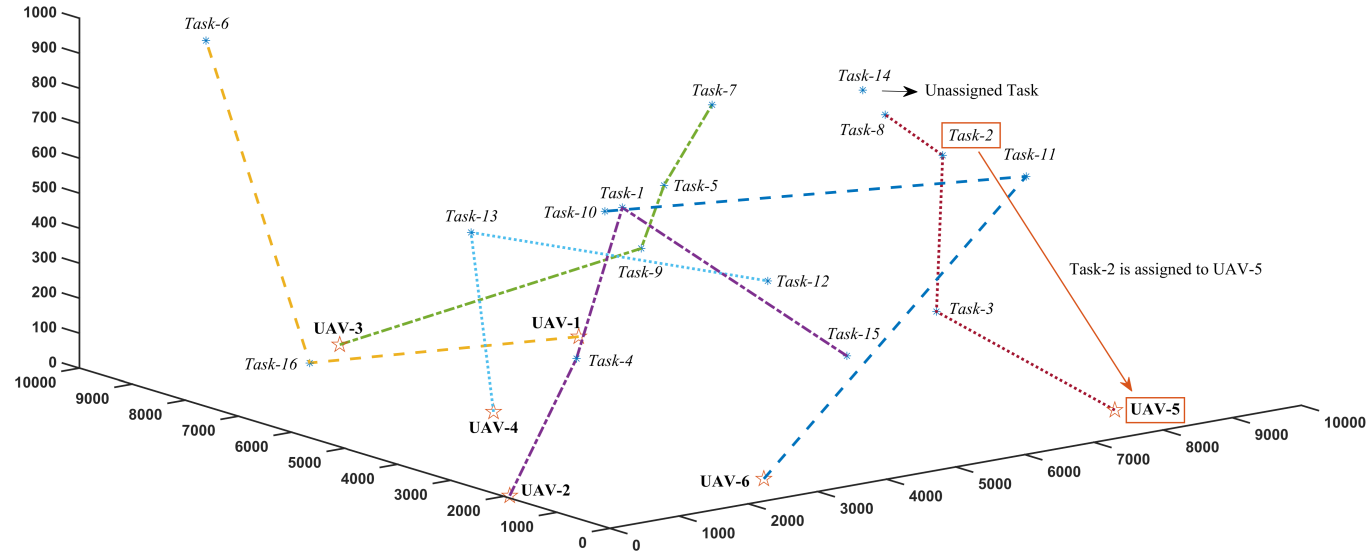

Fig. 6: Static Allocation for Simulation Scenario 3

TABLE III: Task Duration Variation at the start of Task Execution

			Static Task Allocation			Dynamic Task Allocation		
UAV # 5	7251 s		**No. of Iterations**		7	**No. of Iterations**	2	
	UAV Battery Time	**Task**	T_3	T_2	T_8	T_3	T_8	
		Arrival Time	103.9	231.5	444.3	103.9	340.1	
		Priority	0.4756	0.5382	0.8814	0.4756	0.8814	
		Deadline	111	257	520	111	520	
UAV # 6	17295 s	**Task**	T_{11}	T_{10}		T_2	T_{11}	T_{10}
		Arrival Time	111.9	401.2		227	533.9	823.1
	UAV Battery Time	**Priority**	0.7875	0.9860		0.5382	0.7875	0.9860
		Deadline	552	871		257	552	871

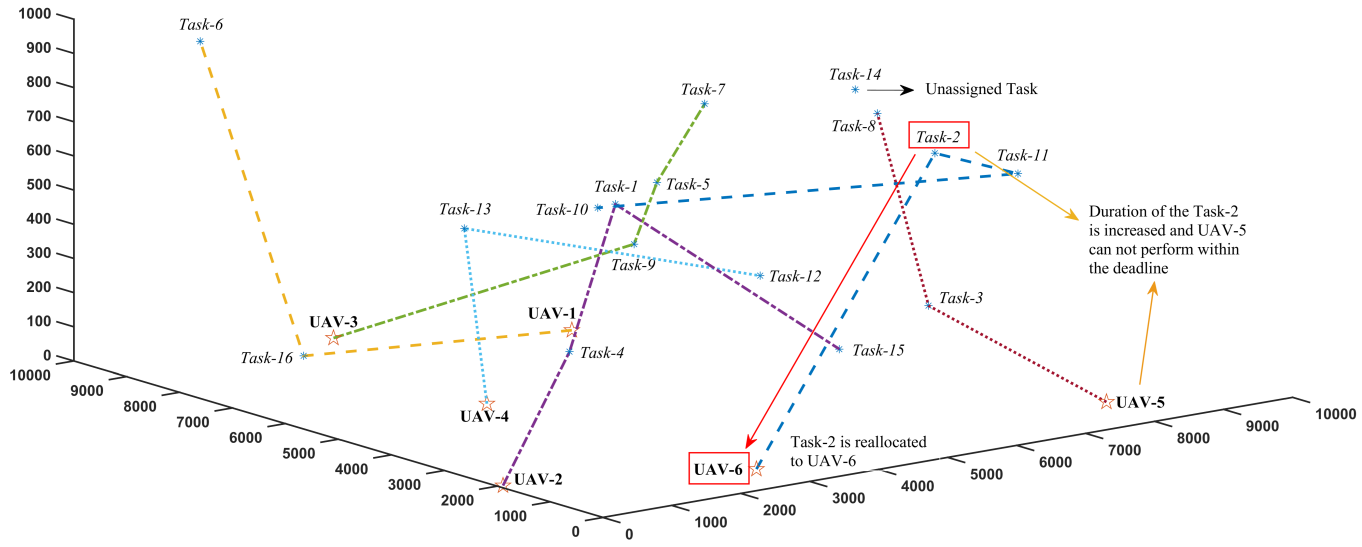

Fig. 7: Extension in Task Duration during Task Execution

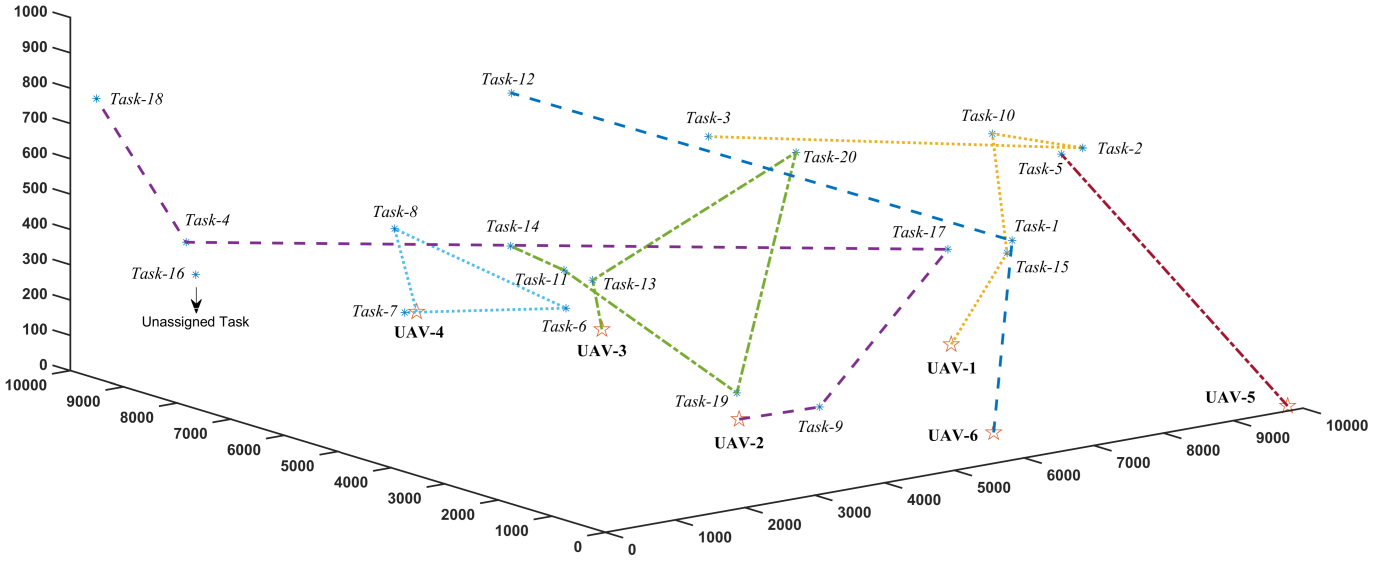

Fig. 8: Static Allocation for Simulation Scenario 5

respectively. At T_d of $28s$, UAV u_2 is moving towards its first task location; therefore, all of its tasks are re-assignable except the first one. With the deletion of t_4, task t_{16} is assigned to UAV u_2 as shown in Figure 9.

F. Simulation Scenario 6: Introduction of a New UAV in Multi-UAV System

The 20 tasks are generated for assignment to 6 UAVs, and the 19 tasks are assigned to 6 UAVs in 12 iterations, as shown in Figure 10. From Figure 10 and Table V, it is clear that t_{19} is a high-priority task with a short deadline as compared to t_{11}; therefore, t_{11} could not be assigned to u_3. Based on the available data, i.e., 6 UAV and 12 iterations for static task allocation, the value of T_{cd} is computed as $10s$ and T_{reas} as $24s$ respectively. A new UAV u_7 is introduced in the system as the start of the task execution phase; therefore, T_{ss} is taken as 0, the value of T_d is $34s$ and hence, tasks starting after $34s$ are re-assignable. Since task cost is equal to time to reach the task location and time to perform that task. Since task duration is taken as 100, each task in the task list of all UAVs except the first task is re-assignable. Therefore, considering the best position in the task list, unassigned tasks t_{19} and t_5 initially assigned to u_1 are reassigned to u_7, respectively, as shown in Figure11.

VI. Conclusion

This paper explores the problem of task allocation in a scenario where multiple UAVs are involved in SAR operations under dynamic conditions. The dynamic events considered in this study include variations in the duration and deadline of tasks, UAV faults, and the deletion or addition of new UAVs to the system to improve the performance of the CDPI

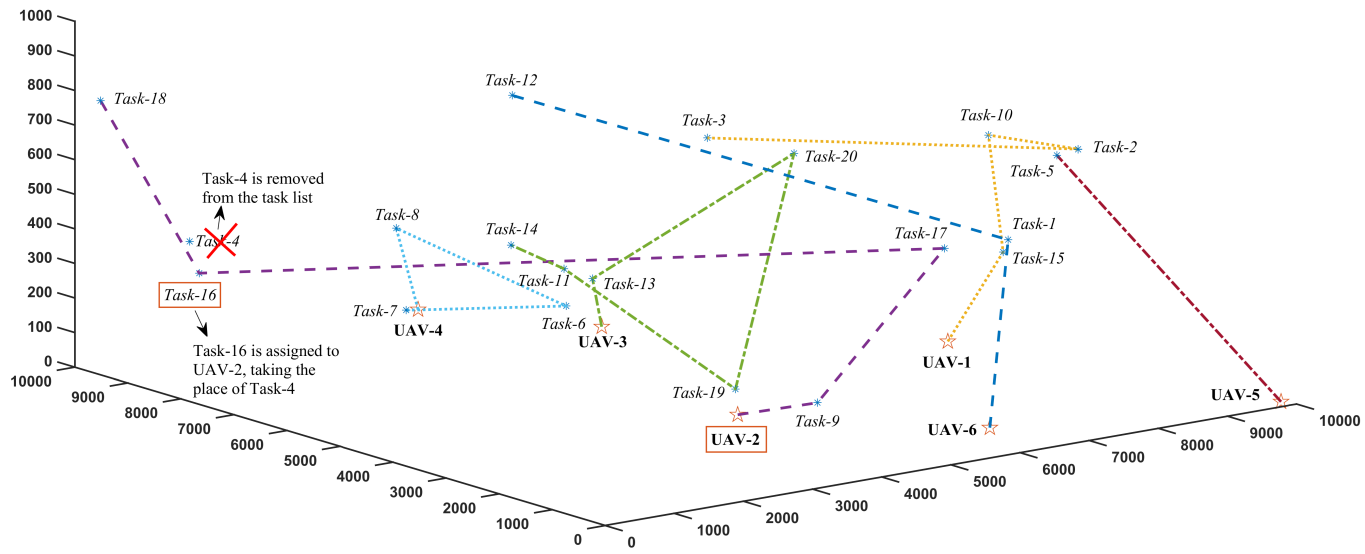

Fig. 9: Deletion of a Task during Task Execution

TABLE IV: Task Deletion at the start of Task Execution

UAV # 2	15799 s	Task	Static Task Allocation				Dynamic Task Allocation			
			No. of Iterations		9		No. of Iterations		2	
	UAV Battery Time		T_9	T_{17}	T_4	T_{18}	T_9	T_{17}	T_{16}	T_{18}
		Arrival Time	30.7	185.4	578.2	714.9	30.7	185.4	573.4	715.2
		Priority	0.3218	0.4620	0.8482	1	0.3218	0.4620	0.8300	1
		Deadline	465	475	616	777	465	475	673	777

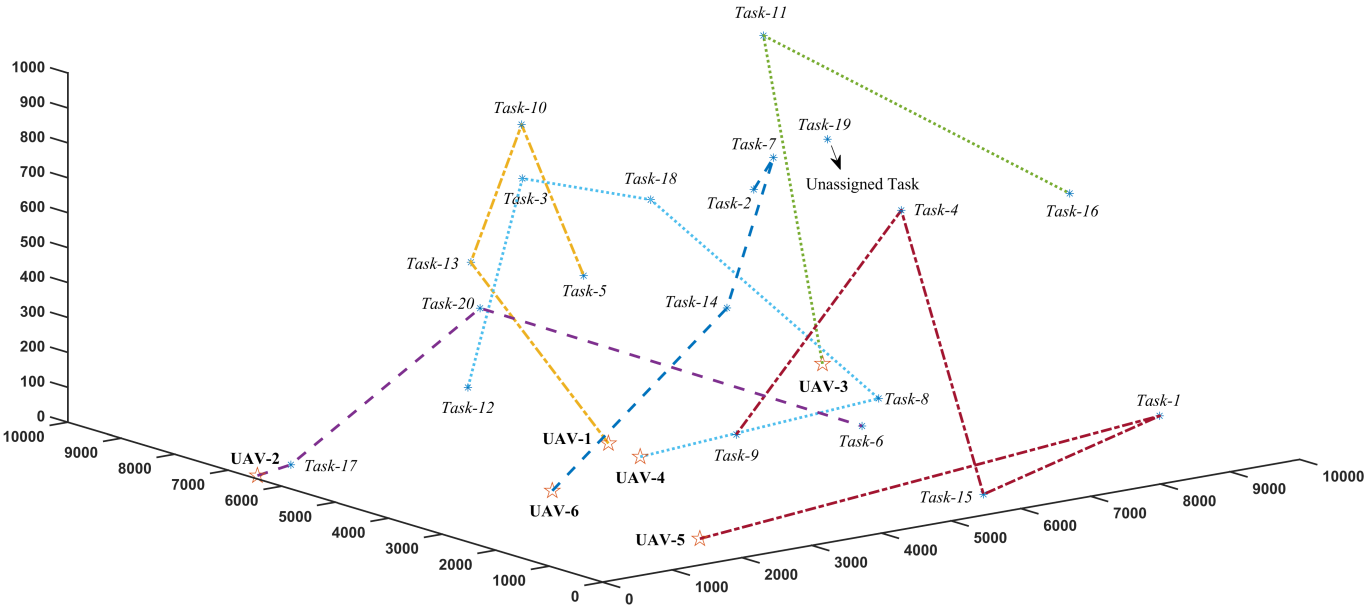

Fig. 10: Static Allocation for Simulation Scenario 6

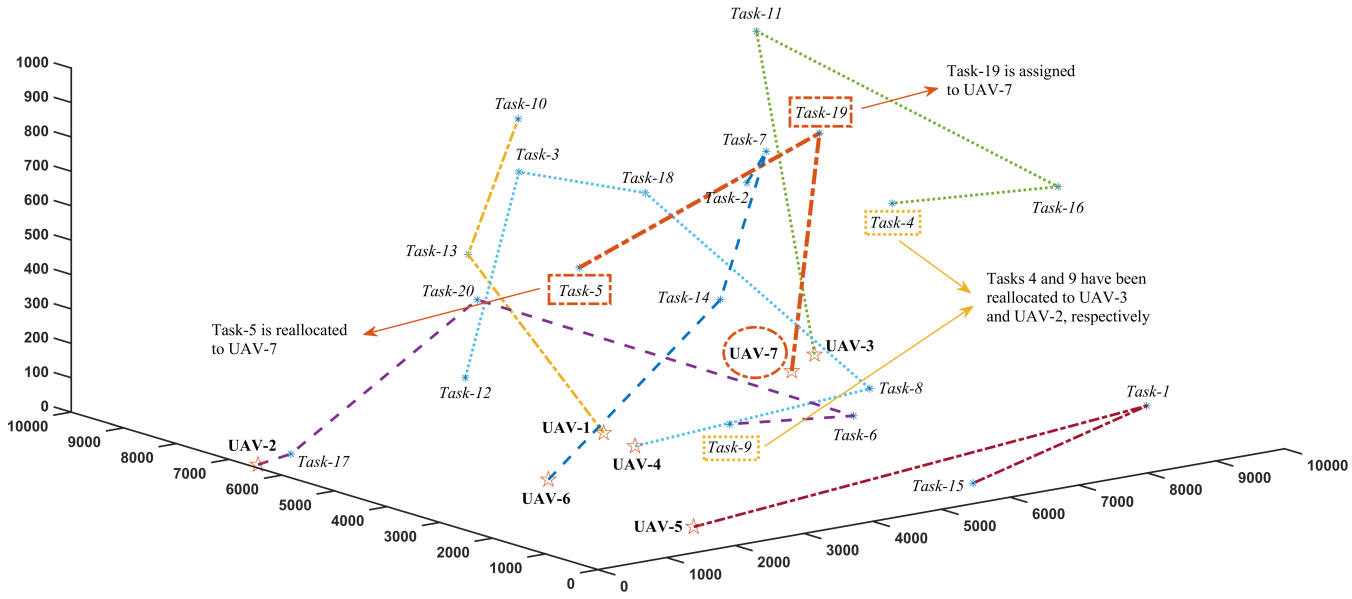

Fig. 11: Insertion of a new UAV during Task Execution

TABLE V: Inclusion of a New UAV at the start of Task Execution

UAV # 1	4302 s		**Static Task Allocation**			**Dynamic Task Allocation**	
		Task	**No. of Iterations**		12	**No. of Iterations**	3
	UAV Battery Time		T_{13}	T_{10}	T_5	T_{13}	T_{10}
		Arrival Time	109	233	387.6	109	233
		Priority	0.6668	0.6364	0.8199	0.6668	0.6364
		Deadline	918	542	432	918	542
UAV # 7	11000 s	**Task**	**UAV inserted during task execution**			T_{19}	T_5
		Arrival Time				25.8	235.8
	UAV Battery Time	**Priority**				0.6527	0.8199
		Deadline				45	432

algorithm. The simulation results show that when a UAV fails, its assigned tasks are automatically reassigned to other UAVs while ensuring that all constraints are satisfied. Additionally, the proposed ECDPI algorithm can handle variations in task duration and deadline. Furthermore, the ECDPI algorithm is also capable of successfully dealing with dynamic events, such as adding a new UAV to the system or deleting a task. In the future, the study will consider the following aspects:

1) Dynamic events are handled separately in the presented simulation. The simulation needs to be enhanced to cater to multiple dynamic events.
2) In the simulations presented, only one instance of a dynamic situation is considered in each simulation. This means that only one task is deleted or one UAV is included in the system. In order to handle multiple instances of a dynamic event, the simulation needs to be enhanced.
3) The proposed ECDPI algorithm requires enhancements to accommodate dynamic events such as changes in battery limit, task priority, etc., during task execution.

Author Contributions: Conceptualization, Rahim Ali Qamar, Sajjad A. Ghauri, and Mubashar Sarfraz; Methodology, Rahim Ali Qamar, Sajjad A. Ghauri, Mubashar Sarfraz and Nauman Anwar Baig; Software, Rahim Ali Qamar, Sajjad A. Ghauri, Mubashar Sarfraz and Nauman Anwar Baig; Validation, Sajjad A. Ghauri, Mubashar Sarfraz, Tanweer Ahmad Cheema and Nauman Anwar Baig; Formal analysis, Rahim Ali Qamar, Sajjad A. Ghauri, Mubashar Sarfraz; Investigation, Rahim Ali Qamar, Sajjad A. Ghauri, and Tanweer Ahmad Cheema; Resources, Sajjad A. Ghauri, Mubashar Sarfraz and Nauman Anwar Baig; Data curation, Rahim Ali Qamar, Sajjad A. Ghauri, Mubashar Sarfraz, and Nauman Anwar Baig; Writing—original draft preparation, Rahim Ali Qamar, Sajjad A. Ghauri, Mubashar Sarfraz, Tanweer Ahmad Cheema and Nauman Anwar Baig; Writing—review and editing, Rahim Ali Qamar, Sajjad A. Ghauri, Mubashar Sarfraz, Tanweer Ahmad Cheema and Nauman Anwar Baig; Visualization, Sajjad A. Ghauri and Mubashar Sarfraz; Supervision, Sajjad A. Ghauri; All authors have read and agreed to the published version of the manuscript.

Funding: The conducted research is slated for publication through the Robert Gordon University Journal Fee Support Program.

Data Availability: Not applicable

Conflicts of Interest: The authors declare no conflict of interest.

Acknowledgment: The conducted research is slated for publication through the Robert Gordon University, Scotland, Journal Fee Support Program.

REFERENCES

[1] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, 2017.

[2] J. García-Pulido, G. Pajares, and S. Dormido, "Uav landing platform recognition using cognitive computation combining geometric analysis and computer vision techniques," *Cognitive Computation*, pp. 1–21, 2022.

[3] F. Zhao, Y. Zeng, G. Wang, J. Bai, and B. Xu, "A brain-inspired decision making model based on top-down biasing of prefrontal cortex to basal ganglia and its application in autonomous uav explorations," *Cognitive Computation*, vol. 10, pp. 296–306, 2018.

[4] X. Liu, Z. Li, N. Zhao, W. Meng, G. Gui, Y. Chen, and F. Adachi, "Transceiver design and multihop d2d for uav iot coverage in disasters," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1803–1815, 2018.

[5] G. Wu, W. Pedrycz, H. Li, M. Ma, and J. Liu, "Coordinated planning of heterogeneous earth observation resources," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 1, pp. 109–125, 2015.

[6] S. Zhang and J. Liu, "Analysis and optimization of multiple unmanned aerial vehicle-assisted communications in post-disaster areas," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 049–12 060, 2018.

[7] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative robots and sensor networks 2015*, pp. 31–51, 2015.

[8] 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*, vol. 7, pp. 5853–5864, 2018.

[9] W. Shen, L. Wang, and Q. Hao, "Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 4, pp. 563–577, 2006.

[10] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2012.

[11] S. Xie, A. Zhang, W. Bi, and Y. Tang, "Multi-uav mission allocation under constraint," *Applied Sciences*, vol. 9, no. 11, p. 2184, 2019.

[12] 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 transactions on cybernetics*, vol. 46, no. 4, pp. 902–915, 2015.

[13] R. A. Qamar, M. Sarfraz, A. Rahman, and S. A. Ghauri, "Multi-criterion multi-uav task allocation under dynamic conditions," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 9, p. 101734, 2023.

[14] M. Akram, A. N. Al-Kenani, J. C. R. Alcantud *et al.*, "Group decision-making based on the vikor method with trapezoidal bipolar fuzzy information," *Symmetry*, vol. 11, no. 10, p. 1313, 2019.

[15] M. Yang, W. Bi, A. Zhang, and F. Gao, "A distributed task reassignment method in dynamic environment for multi-uav system," *Applied Intelligence*, vol. 52, no. 2, pp. 1582–1601, 2022.

[16] A. Adeel, M. Gogate, S. Farooq, C. Ieracitano, K. Dashtipour, H. Larijani, and A. Hussain, "A survey on the role of wireless sensor networks and iot in disaster management," *Geological disaster monitoring based on sensor networks*, pp. 57–66, 2019.

[17] E. Osaba, J. Del Ser, A. D. Martinez, and A. Hussain, "Evolutionary multitask optimization: a methodological overview, challenges, and future research directions," *Cognitive Computation*, vol. 14, no. 3, pp. 927–954, 2022.

[18] C. Wei, K. V. Hindriks, and C. M. Jonker, "Dynamic task allocation for multi-robot search and retrieval tasks," *Applied Intelligence*, vol. 45, pp. 383–401, 2016.

[19] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous hedonic game for task allocation in a large-scale multiple agent system," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1534–1548, 2018.

[20] A. Whitbrook, Q. Meng, and P. W. Chung, "Reliable, distributed scheduling and rescheduling for time-critical, multiagent systems," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 732–747, 2017.

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

[22] X. Chen, P. Zhang, G. Du, and F. Li, "A distributed method for dynamic multi-robot task allocation problems with critical time constraints," *Robotics and Autonomous Systems*, vol. 118, pp. 31–46, 2019.

[23] U. Baroudi, M. Alshaboti, A. Koubaa, and S. Trigui, "Dynamic multi-objective auction-based (dymo-auction) task allocation," *Applied Sciences*, vol. 10, no. 9, p. 3264, 2020.

[24] Q. Wang and X. Mao, "Dynamic task allocation method of swarm robots based on optimal mass transport theory," *Symmetry*, vol. 12, no. 10, p. 1682, 2020.

[25] W. Dai, H. Lu, J. Xiao, Z. Zeng, and Z. Zheng, "Multi-robot dynamic task allocation for exploration and destruction," *Journal of Intelligent & Robotic Systems*, vol. 98, pp. 455–479, 2020.

[26] X. Wu, Z. Gao, S. Yuan, Q. Hu, and Z. Dang, "A dynamic task allocation algorithm for heterogeneous uuv swarms," *Sensors*, vol. 22, no. 6, p. 2122, 2022.

[27] S. Choudhury, J. K. Gupta, M. J. Kochenderfer, D. Sadigh, and J. Bohg, "Dynamic multi-robot task allocation under uncertainty and temporal constraints," *Autonomous Robots*, vol. 46, no. 1, pp. 231–247, 2022.

[28] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International journal of robotics research*, vol. 23, no. 9, pp. 939–954, 2004.

[29] R. A. Qamar, M. Sarfraz, S. A. Ghauri, and A. Mahmood, "Trmaxalloc: Maximum task allocation using reassignment algorithm in multi-uav system," *Computer Communications*, vol. 206, pp. 110–123, 2023.

[30] Z. Gao, R. Y. Liang, and T. Xuan, "Vikor method for ranking concrete bridge repair projects with target-based criteria," *Results in Engineering*, vol. 3, p. 100018, 2019.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- RahimFINAL.rar