

Article

Comparative Analysis of Centralized and Distributed Multi-UAV Task Allocation Algorithms: A Unified Evaluation Framework

Yunze Song [†], Zhexuan Ma [†], Nuo Chen [†], Shenghao Zhou [†] and Sutthiphong Srigrarom ^{*†} 

Mechanical Engineering, National University of Singapore, 5A Engineering Drive 1, Singapore 117411, Singapore

^{*} Correspondence: spot.srigrarom@nus.edu.sg[†] These authors contributed equally to this work.

Abstract

Unmanned aerial vehicles (UAVs), commonly known as drones, offer unprecedented flexibility for complex missions such as area surveillance, search and rescue, and cooperative inspection. This paper presents a unified evaluation framework for the comparison of centralized and distributed task allocation algorithms specifically tailored to multi-UAV operations. We first contextualize the classical assignment problem (AP) under UAV mission constraints, including the flight time, propulsion energy capacity, and communication range, and evaluate optimal one-to-one solvers including the Hungarian algorithm, the Bertsekas ϵ -auction algorithm, and a minimum cost maximum flow formulation. To reflect the dynamic, uncertain environments that UAV fleets encounter, we extend our analysis to distributed multi-UAV task allocation (MUTA) methods. In particular, we examine the consensus-based bundle algorithm (CBBA) and a distributed auction 2-opt refinement strategy, both of which iteratively negotiate task bundles across UAVs to accommodate real-time task arrivals and intermittent connectivity. Finally, we outline how reinforcement learning (RL) can be incorporated to learn adaptive policies that balance energy efficiency and mission success under varying wind conditions and obstacle fields. Through simulations incorporating UAV-specific cost models and communication topologies, we assess each algorithm's mission completion time, total energy expenditure, communication overhead, and resilience to UAV failures. Our results highlight the trade-off between strict optimality, which is suitable for small fleets in static scenarios, and scalable, robust coordination, necessary for large, dynamic multi-UAV deployments.

Keywords: assignment problem (AP); multi-UAV task allocation (MUTA); multi-agent task allocation (MATA); consensus-based bundle algorithm (CBBA); Bertsekas ϵ -auction algorithm; Hungarian algorithm; one-to-one assignments; centralized and distributed algorithms; evaluation frameworks

Academic Editors: Jingjing Wang,
Yibo Zhang and Qi Li

Received: 20 May 2025

Revised: 18 July 2025

Accepted: 18 July 2025

Published: 28 July 2025

Citation: Song, Y.; Ma, Z.; Chen, N.; Zhou, S.; Srigrarom, S. Comparative Analysis of Centralized and Distributed Multi-UAV Task Allocation Algorithms: A Unified Evaluation Framework. *Drones* **2025**, *9*, 530. <https://doi.org/10.3390/drones9080530>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

UAVs and drones have rapidly proliferated in both the civilian and military domains due to their mobility, flexibility, and ability to operate as coordinated teams [1]. Modern multi-UAV systems undertake missions ranging from urban surveillance and environmental monitoring to search and rescue and delivery logistics [2]. By deploying multiple UAVs cooperatively, these systems can cover large areas and perform tasks in parallel, achieving far greater efficiency than any single UAV. For example, a fully autonomous swarm of palm-sized drones has been shown to navigate a dense forest, requiring real-time coordination to avoid obstacles and inter-drone collisions [3]. This showcases the potential

of multi-UAV teams to operate in complex, previously inaccessible environments, but also underscores the need for robust task allocation and coordination strategies to manage such missions.

From a broad perspective, the MATA problem addresses how to assign a set of tasks among multiple autonomous agents (robots or UAVs) to optimize the overall performance [4]. In the specific context of multi-robot systems, MUTA refers to the coordinated process of assigning mission tasks to a team of drones. Effective task allocation becomes critical when no single UAV has the capability or capacity to accomplish all tasks alone—a common situation in complex missions such as wide-area surveillance or disaster response. In these scenarios, tasks must be intelligently divided among UAVs according to each drone's capabilities (e.g., sensor payload, endurance) and the mission requirements. Recent work has emphasized that multi-UAV missions introduce unique challenges not seen in other multi-robot settings. For instance, persistent monitoring tasks with UAVs must contend with time-varying environmental states, limited flight endurance (propulsion energy capacity), and constrained communication ranges between aerial units [2]. Failing to account for these factors can significantly degrade mission performance. This is highlighted in a distributed persistent monitoring approach, where dynamic task reallocation is continually needed to handle evolving information and keep data fresh [2]. In general, as the number of drones and tasks grows, the task allocation problem quickly becomes combinatorially complex and intractable to solve optimally in real time (the assignment problem is NP-hard in many formulations). This has motivated a spectrum of approaches, from classical optimization to market-based auctions and machine learning, which yield sufficient solutions within practical time ranges for multi-UAV applications.

A fundamental design decision for MUTA is the coordination architecture, which can be roughly divided into centralized and distributed schemes. In a centralized architecture, a single control node (either an onboard designated UAV or a ground control station) is responsible for aggregating global state information, computing the optimal or near-optimal tasking assignment for the entire fleet, and issuing commands accordingly [5]. The advantages of this approach are that it guarantees global optimality under certain assumptions, simplifies system-wide monitoring, and facilitates the integration of complex optimization algorithms [6,7]. However, centralized solutions have some key drawbacks, including single points of failure, large communication overheads (which scale poorly with increasing team sizes), reduced robustness in dynamic or competitive environments, and latency issues due to non-real-time data aggregation and decision computation [8,9].

Conversely, distributed architectures decentralize the decision-making process across individual UAVs or local clusters. Each agent independently computes task assignments based on locally available information and peer-to-peer communications, often using consensus algorithms or market-based negotiation mechanisms [10,11]. Distributed schemes inherently improve scalability, reduce the reliance on high-bandwidth communication links, and increase fault tolerance by eliminating central points of failure [2,12]. Nevertheless, decentralized approaches may yield suboptimal allocation due to incomplete global information, require complex coordination protocols to reach consensus, and can be susceptible to communication delays and inconsistent state knowledge among agents [6,7].

Centralized algorithms like the Hungarian assignment method can produce globally optimal allocations under complete information [13]. However, this optimality comes at the cost of heavy communication and a single point of failure. Every UAV must continuously share its status (position, fuel, etc.) with the central controller, which can overload networks and create bottlenecks [13]. Moreover, if the central node fails or loses connectivity, the entire mission could be jeopardized [13]. By contrast, distributed (or decentralized) approaches avoid any one single leader; drones make decisions locally through peer-to-peer

communication or consensus. This yields far greater robustness to failures and communication dropouts and tends to scale better as the team grows [13]. The trade-off is that distributed methods may sacrifice some optimality and coordination precision. Without a global view, drones might make locally rational choices that are suboptimal globally—leading to conflicts (e.g., multiple UAVs choosing the same task) or idle resources [13]. For example, market-based auction algorithms allow UAVs to bid for tasks and iteratively reach an assignment; these are highly scalable and flexible, but they typically find near-optimal rather than truly optimal allocations. Recent research has explored hybrid architectures (sometimes called hierarchical or hybrid coordination), where elements of centralized planning are combined with distributed execution to achieve the best of both worlds [2]. Overall, the choice of a centralized vs. distributed strategy has profound implications for MUTA performance, especially in contested or communication-limited environments.

Another important consideration is the mode of cooperation among UAVs. In some missions, tasks are independent and can be simply divided among individual drones. In others, such as transporting a heavy payload or in a wide-area search, forming coalitions or sub-teams of UAVs for a task can be advantageous. Recent works have begun treating MUTA as a coalition formation problem, using game-theoretic frameworks [1]. In such formulations, UAVs decide not only which tasks to take but also whether to team up with other drones to execute a task jointly. Compared to a single UAV operating alone, appropriately grouping UAVs into coalitions can dramatically enhance the task efficiency for complex objectives [1]. These coalition-based approaches introduce additional layers of decision-making (e.g., how to split rewards among UAVs, how to ensure stable cooperation within a coalition), but they are highly relevant for missions like disaster response, where tasks might demand multiple drone capabilities simultaneously. In summary, the background of MUTA is rooted in classic multi-agent assignment problems but extended by the distinct constraints and possibilities of drone technology, which include high mobility, energy limits, wireless communication, and the ability to coordinate in swarms or formations. This paper is motivated by the need to systematically evaluate how different task allocation algorithms (centralized optimal, distributed heuristic, learning-based, etc.) perform in coordinating UAV teams, given the growing deployment of drone swarms in real-world applications.

1.1. Challenges in Task Allocation

Effectively allocating tasks among multiple UAVs is difficult due to several overlapping challenges that must be addressed simultaneously. We highlight some of the key challenges below, particularly focusing on those that are salient in multi-UAV contexts.

- **Computational Complexity and Scalability**

Optimal MUTA is an inherently combinatorial problem that becomes intractable at scale. Even a simplified assignment scenario (where each drone performs one task at a time) can be mapped to an NP-hard assignment problem when extended with realistic constraints [14]. The number of possible task-to-UAV allocations grows exponentially with the team size and task count, making a brute-force search impractical [15,16]. This is exacerbated in dynamic scenarios, where the assignment may need to be recomputed frequently as new tasks appear or conditions change. Thus, algorithms must be designed to scale gracefully. A related issue is communication scalability: a centralized allocator requires gathering cost/utility information from all UAVs, incurring an $O(m \cdot n)$ communication overhead for m drones and n tasks each cycle [15,16]. This can quickly tax wireless bandwidth and processing resources in a large drone swarm [15,16]. Decentralized approaches mitigate the single-channel bottleneck, but they still require agents to exchange messages to reach a consensus on

assignments. Without careful management, the message complexity can explode as the fleet grows. In sum, ensuring that task allocation methods remain computationally and communicatively efficient for large multi-UAV teams is a primary concern [15].

- **Optimality vs. Real-Time Responsiveness**

There is an inherent trade-off between solution optimality and decision speed in task allocation. On one hand, centrally optimized algorithms (e.g., integer programming solvers or exhaustive search) can, in theory, find the assignment that minimizes the total cost or maximizes the reward for the mission [17]. However, these methods often require significant time and complete, up-to-date information from all drones. In fast-changing environments, the “optimal” allocation may arrive too late to be useful. On the other hand, distributed or greedy heuristic methods can react very quickly to new tasks or events—sometimes in near-constant time by local computation—but they trade away guarantees of optimality [15,16]. A task allocation that is considered optimal at one moment may become suboptimal moments later as UAV positions and task states evolve. Many distributed schemes (e.g., auctions, consensus algorithms) prioritize adaptability and low communication, accepting a slight loss in solution quality. The challenge is to find a balanced approach that yields high-quality assignments within the tight timing constraints of UAV missions. For example, a multi-agent RL approach was used to coordinate heterogeneous UAVs, enabling on-the-fly task reallocation with minimal delays and maintaining the performance even as the team size scaled to dozens of drones [18]. Such learning-based or anytime algorithms aim to continuously improve task allocations and approach optimality over time, without stalling the mission with lengthy computations.

- **Dynamic and Uncertain Environments**

Multi-UAV operations are often conducted in dynamic environments with significant uncertainty. In real-world deployments like disaster response, military missions, or environmental surveillance, tasks can arrive unpredictably (e.g., a new fire outbreak to monitor, a new target spotted) and task requirements can change as scenarios unfold. A previously optimal task assignment can be invalidated by the appearance of a high-priority new task or the sudden inaccessibility of an area [11,19]. Therefore, task allocation algorithms must handle dynamic changes—they should recompute or adjust assignments efficiently as new tasks appear or old tasks evolve. This might involve techniques like continuous replanning, rolling horizon optimization, or agents making local decisions based on current observations. Uncertainty further complicates planning: UAVs may have imprecise information about task rewards or their own states (due to sensor noise or communication delays). Travel times or task completion times might be stochastic. Algorithms thus need to be robust to incomplete and noisy information, perhaps by anticipating changes or quickly correcting assignments when surprises occur. For example, some approaches incorporate safety margins or periodic reallocation triggers to cope with uncertainty [11,19]. Overall, the dynamic nature of UAV missions demands task allocation methods that are reactive and adaptive, rather than one-shot static plans.

- **Communication Constraints**

Reliable communication is a persistent challenge for distributed UAV swarms. Drones often operate in environments with limited bandwidth, intermittent connectivity, or strict line-of-sight constraints (e.g., within buildings, underground, or over long distances). High communication demands can severely degrade a task allocation algorithm’s performance. For instance, auction-based methods where UAVs continuously broadcast bids for tasks may work well in a lab or simulation, but, in the field, these messages can be lost or delayed, leading to suboptimal or failed allocations [20,21].

Studies have shown that algorithms that perform well with perfect networking can break down when packet loss or latency is introduced. The challenge is to design communication-efficient task allocation algorithms that minimize the amount of information exchanged or that can tolerate delays and dropouts. Some recent advances leverage consensus protocols that require only neighbor-to-neighbor communication and can converge despite asynchronous updates or occasional message loss [20,21]. Another strategy keeps communication local by structuring the team into clusters or hierarchies, limiting how far messages must travel [20,21]. Ultimately, as UAVs often operate on wireless ad hoc networks, resilience to communication disruptions is a key requirement for any practical multi-UAV coordination strategy.

- **Robustness to Agent Failures**

In critical missions (e.g., search and rescue, exploration of hazardous areas), it is realistic to expect that some UAVs might fail, lose power, or drop out due to collisions, propulsion energy depletion, or other hazards. A robust task allocation system must dynamically reassign the tasks of a failed drone to others so that the mission can continue with minimal degradation. Centralized systems are particularly vulnerable: if the central controller UAV goes down, it can cripple the whole team's coordination [22,23]. Decentralized approaches inherently distribute decision-making, which improves fault tolerance—the failure of one drone (even if it is a leader for some tasks) can be detected and compensated for by others without a single point of collapse [22]. For example, a recent framework called MAGNNET combined multi-agent deep learning with graph neural networks to enable fully decentralized task allocations; it demonstrated improved scalability and fault tolerance in simulations, continuing to perform well even as agents were removed or communications were intermittently lost [13,22,23]. Another aspect of robustness is handling sudden surges in task load. If a large number of new tasks appears at once (e.g., multiple disaster sites in an earthquake), the allocation algorithm should gracefully redistribute the workload and avoid overloading any single UAV [22,23]. In summary, ensuring graceful degradation and quick recovery in the face of drone failures or other disruptions is crucial for MUTA. Techniques like redundant task assignments, peer monitoring (drones watching each other's status), and on-the-fly replanning contribute to a robust allocation system.

- **Fairness and Load Balancing**

While not always a major technical requirement, fairness in task distribution can impact the long-term effectiveness of a multi-UAV team. If one or two drones consistently handle the largest workloads or the most strenuous tasks, they may exhaust their batteries or even suffer faster wear and tear, while other UAVs remain underutilized. In human–robot or mixed-initiative teams, unfair task allocation can also affect human trust and team morale [20,24]. Therefore, some task allocation strategies incorporate load-balancing mechanisms. For instance, each UAV's current workload or recent task history can be factored into new assignment decisions. There is often a tension between purely cost-optimal assignments (which might overburden the most capable UAV) and fair assignments that spread tasks more evenly. Metrics have been proposed to explicitly measure how balanced an allocation is across the fleet. In UAV networks, a balanced approach might extend the overall mission duration by preventing any single drone from draining its propulsion energy too quickly. The challenge for the algorithm is to achieve good performance while avoiding severe workload imbalances. In practice, modest fairness constraints (like avoiding assigning a new task to a UAV that is already at capacity if an alternative UAV is available) can significantly improve the team's resilience and longevity [20,24].

In summary, the task allocation problem for multi-UAV systems lies at the intersection of combinatorial optimization, real-time systems, and networked robotics. An effective solution must navigate the complexities of scale, dynamic changes, limited communication, and the practical realities of drone operations (like failures and propulsion energy limits). The above challenges often conflict with each other—improving one aspect (say, making the solution more optimal) might worsen another (like the computation time or communication load). The goal for researchers is to design algorithms that balance these trade-offs, delivering reliable and efficient task assignments for drone swarms in real-world conditions.

1.2. Research Gaps and Objectives

Thanks to intensive research over the past decade, a wide range of algorithms now exist for MUTA [11,15]. However, important gaps remain in our understanding of how different approaches compare and which are best suited for certain scenarios. Early research in MUTA tended to develop bespoke algorithms for specific mission types, without providing a broader comparative perspective. For example, one study might propose a new allocation method tailored to a surveillance mission, while another study might design a centralized planner for a delivery drone network, but each is evaluated in isolation, under its own assumptions and environment [25,26]. This means that, until recently, we lacked “horizontal” comparisons across algorithm families [9,27]. Researchers seldom directly compare a centralized optimal method against a distributed heuristic under identical conditions, leaving practitioners unsure of how each would perform outside of their respective case studies [12,28,29].

In the literature, most comparative studies have been narrow in scope. Some works focus only on one class of algorithms—for instance, comparing different distributed algorithms with each other (assuming a decentralized setting) while excluding centralized or learning-based approaches from the analysis [30,31]. Other works may compare a classic centralized algorithm against a few heuristics, but these often neglect realistic issues like communication delays or dynamic task arrivals in their evaluations [9,31]. In short, there has been a lack of unified benchmarking. As Alqefari and Menai (2025) point out in their recent survey, studies often evaluate task allocation methods under varying assumptions and metrics, making it difficult to draw general conclusions or best practices [9]. For example, one paper might measure success purely by the total task completion time, while another prioritizes the communication overhead or success rate, complicating direct comparison [12,20]. No consensus has emerged on which algorithmic paradigm is “best” for multi-UAV coordination, partly because each has advantages under certain conditions (e.g., centralized for static environments vs. distributed for highly dynamic situations) [15,27].

Another gap is the relative scarcity of evaluations that combine multiple real-world challenges in one scenario. Many experiments simplify the problem to isolate one factor, e.g., testing algorithms in a static environment with no new tasks, assuming perfect communication, or having a moderate team size [9,9]. In reality, a multi-UAV mission may involve all these challenges at once—a large team in a dynamic environment with intermittent communication and potential UAV failures [2,30]. Few studies have stress-tested algorithms under such comprehensive conditions [32]. Notably, most prior benchmarks either use static or mildly dynamic settings [27]. There are few datasets or simulations that consider high-frequency task influx, large-scale surges in task load, and agent dropouts together across competing algorithms [26]. As a consequence, we do not fully know how, for example, a market-based method holds up when both communication is lossy and tasks arise rapidly, compared to how a learning-based method or an optimization-based method would cope in the same stressful scenario [11,32].

The objective of our work is to fill these gaps by providing a unified evaluation of major task allocation algorithms for multi-UAV systems. We aim to rigorously compare a representative set of algorithms—spanning centralized, distributed, and learning-based approaches—under identical conditions and across a variety of scenarios. By doing so, we seek to answer the following questions: Which algorithms handle highly dynamic task streams best? How much optimality do distributed methods lose in exchange for scalability and robustness? At what team size or task load does a centralized method break down? Moreover, we incorporate practical factors (like communication limits and UAV failures) into the evaluation, to examine each method’s robustness beyond idealized assumptions. Ultimately, the goal is not to declare a single “winner” but to chart the trade-offs: identifying which algorithm is preferable under what circumstances. This type of comprehensive comparison can provide valuable guidance to practitioners in choosing a solution for a given multi-UAV application. It can also highlight the strengths of each approach and reveal areas where improvement or hybridization is needed (for example, if a learning-based method performs well generally but struggles with fairness or specific constraint handling, which points to future research directions).

In summary, while the toolkit of MUTA algorithms has grown, the lack of unified comparisons and integrated testing has left a gap in understanding when and why to choose one method over another. Our research addresses this by benchmarking diverse algorithms within a common framework, thereby providing insights into their relative performance, limitations, and suitability for various multi-UAV mission profiles. The findings are intended to both inform real-world deployments (by matching algorithms to use cases) and pinpoint where further research is needed (such as improving an algorithm’s scalability or adaptability).

1.3. Contributions of This Paper

This paper presents a systematic study of MATA with the following contributions.

1. Comprehensive Comparison Framework

We propose a unified framework for the comparison of centralized and distributed task allocation methods (e.g., Hungarian, auction, CBBA, and auction 2-opt refinement algorithms) in both static and dynamic scenarios. The implementation in this study is available on GitHub: <https://github.com/YunzeSong/Multi-UAVs-Task-Allocation-Algorithms.git> (accessed on 18 April 2025).

2. Reproducible Simulation Environment

A simulation framework is developed in Python with standardized interfaces to evaluate algorithms across various scales (small, medium, large) and dynamic conditions.

3. Multi-Dimensional Evaluation Metrics

We establish evaluation metrics covering the total cost, computation time, convergence, communication overhead, robustness, and scalability, supported by clear visualizations.

4. Analysis of Centralized vs. Distributed Trade-Offs

This paper analyzes the trade-offs between global optimality and scalability/robustness, especially under dynamic task arrivals and agent failures.

2. Literature Review

2.1. Centralized Task Allocation Algorithms

Hungarian Algorithm (centralized) refers to Algorithm 1. The Hungarian algorithm is a classical optimization algorithm for solving assignment problems; it uses graph-theoretic methods to find a globally optimal solution for task assignment in polynomial time [12]. As a centralized approach, it requires centralizing the cost matrices of all tasks and agents to be solved at a single computational node. The Hungarian algorithm is often used as

a benchmark for task allocation and can ensure an allocation scheme that minimizes the total cost. However, it assumes that the tasks are statically known and requires the iterative re-running of the algorithm for dynamic task joining. As the problem size increases, its time complexity is about $O(n^3)$ (where n is the number of tasks/agents), and the computational overhead may become high at very large scales. Moreover, full centrality in multi-agent systems implies a single point of failure risk and strong dependence on communication (global information needs to be transmitted to the central node for processing [33]). The Hungarian algorithm is therefore well suited as a benchmark for optimal solutions for small-scale static scenarios, as well as for evaluating the gaps between other algorithms in terms of cost optimality.

Algorithm 1 Hungarian Algorithm for Optimal Task Assignment

Require: $A = \{a_1, \dots, a_{n_A}\}$, $T = \{t_1, \dots, t_{n_T}\}$, cost function $c : A \times T \rightarrow \mathbb{R}_+$, where

$$c(a_i, t_j) = \begin{cases} \text{TravelTime}(a_i, t_j), & \text{if } a_i \text{ is compatible with } t_j, \\ H, & \text{otherwise} \quad (H \gg 0). \end{cases}$$

Ensure: An assignment set $\mathcal{A}$ satisfying

$$\min_{\mathcal{A}} \sum_{(a,t) \in \mathcal{A}} c(a, t).$$

- 1: Construct the cost matrix $M \in \mathbb{R}^{n_A \times n_T}$ using $c(a_i, t_j)$.
- 2: Solve

$$\min_{(i,j)} \sum M(i, j)$$

via the Hungarian method to obtain optimal index pairs $\{(i, j)\}$.

- 3: Form the assignment set

$$\mathcal{A} = \{(a_i, t_j) \mid (i, j) \text{ optimal and } M(i, j) < H\}.$$

return $\mathcal{A}$.

2.2. Distributed Task Allocation Algorithms

Bertsekas ϵ -Auction Algorithm (centralized/distributed) refers to Algorithm 2. The Bertsekas ϵ -auction algorithm is an allocation algorithm based on a bidding mechanism. The idea is to let agents bid for tasks, which correspond to a continuously adjusted “price”. The algorithm proceeds iteratively: unassigned agents simultaneously bid on the task that they are most interested in, and the task is temporarily assigned to the agent with the highest bid and the price is raised; this continues until all tasks have been assigned. The method is essentially a parallel relaxation of the original assignment problem and can be viewed as an iterative approximation of the duality problem [34]. The Bertsekas ϵ -auction algorithm can be implemented centrally, where an auctioneer (central node) collects the bids of all agents and decides where the task belongs; however, it is also well suited for distributed implementations, where each agent bids autonomously and neighbors exchange price information via multi-hop communication to update their respective decisions [35]. The Bertsekas ϵ -auction algorithm is able to converge under the condition that only local information is utilized, and the final convergence results in the maximum allocation of the total benefit approximating the optimal solution [35]. Since each round of bidding involves local computation and simple communication, the algorithm is easy to parallelize and computationally efficient (up to about $O(n^2)$ for typical implementations). The Bertsekas ϵ -auction algorithm is also suitable for dynamic tasks: when a new task appears, it can be treated as a new auction item, and it is sufficient to allow the relevant agents to partici-

pate in a new round of bidding, without having to compute all allocations from scratch. Thus, the Bertsekas ϵ -auction algorithm offers a method that bridges centralized and decentralized aspects, providing both near-optimal solutions and good scalability through distributed bidding.

Algorithm 2 Bertsekas ϵ -Auction Algorithm for Optimal Task Assignment

Require: $A = \{a_1, \dots, a_{n_A}\}$ (agents), $T = \{t_1, \dots, t_{n_T}\}$ (tasks), a value function

$$v : A \times T \rightarrow \mathbb{R}, \quad v(a_i, t_j) = \begin{cases} -\text{dist}(a_i, t_j), & \text{if } a_i \text{ is compatible with } t_j, \\ -\infty, & \text{otherwise,} \end{cases}$$

a bid increment $\epsilon > 0$ and a maximum iteration limit $K_{\max}$.

Ensure: An assignment set $\mathcal{A} \subset A \times T$ satisfying

$$\min_{\mathcal{A}} \sum_{(a,t) \in \mathcal{A}} [-v(a, t)].$$

- 1: **Initialization:**
- 2: Set task prices: $p(t_j) \leftarrow 0$ for all $t_j \in T$;
- 3: Set each agent's assignment: $S(a_i) \leftarrow \emptyset$ for all $a_i \in A$;
- 4: Let $U \leftarrow A$ be the set of unassigned agents.
- 5: **for** $k = 1, \dots, K_{\max}$ **do**
- 6: **if** $U = \emptyset$ **then**
- 7: **return** $\mathcal{A} = \{(a_i, S(a_i)) \mid S(a_i) \neq \emptyset\}$
- 8: **end if**
- 9: **for all** $a_i \in U$ **do**
- 10: Compute utilities:

$$u(a_i, t_j) = v(a_i, t_j) - p(t_j), \quad \forall t_j \in T.$$

- 11: Let $t^* = \arg \max_{t_j \in T} u(a_i, t_j)$ with $u_{\max} = u(a_i, t^*)$.
- 12: Let

$$u_{2\text{nd}} = \max_{t_j \in T \setminus \{t^*\}} u(a_i, t_j),$$

where if no alternative exists then $u_{2\text{nd}} = -\infty$.

- 13: Compute bid increment: $\delta = u_{\max} - u_{2\text{nd}} + \epsilon$.
- 14: Update price: $p(t^*) \leftarrow p(t^*) + \delta$.
- 15: If t^* is already assigned to agent $a_{i'}$ then

$$\text{Unassign } a_{i'} \text{ by setting } S(a_{i'}) \leftarrow \emptyset, \quad U \leftarrow U \cup \{a_{i'}\}.$$

- 16: Assign a_i to t^* : set $S(a_i) \leftarrow t^*$ and remove a_i from U .
 - 17: **end for**
 - 18: **end for**
 - 19: **return** $\mathcal{A} = \{(a_i, S(a_i)) \mid S(a_i) \neq \emptyset\}$.
-

CBBA (Consensus-Based Bundle Algorithm, distributed) refers to Algorithm 3. CBBA is a state-of-the-art distributed task allocation algorithm that uses a combination of market bidding and locally consistent agreement [33]. In CBBA, each agent greedily constructs a sequence of tasks (a bundle) based on its own capabilities and task benefits, and then the agents communicate with each other to negotiate consistency, exchanging their respective allocation intentions and resolving conflicts (multiple agents choosing the same task) according to a certain set of rules, until all of them agree on the conflict-free allocation of tasks [33]. CBBA has the following characteristics [33]: (1) fully decentralized decision-making with no central node, which is necessary for large-scale teams to avoid centralized computation and communication bottlenecks as the number of agents increases; (2) polynomial time

complexity, where the algorithm grows linearly or polynomially with the number of tasks and agents and thus can still run when the task/agent sizes are large (good scalability); and (3) an ability to deal with complex scenarios, such as each agent performing multiple tasks (time-scaled allocation) and heterogeneous agent and task values, etc., as well as providing guarantees regarding the quality of the solution under certain assumptions (outputs feasible solutions with guaranteed performance [33]). CBBA is suitable for dynamic task scenarios because its iterative process can be triggered to re-execute when the set of tasks changes: when a new task is added or an agent fails, each agent can update its own sequence of tasks and communicate again to reach a new consensus. This locally adaptive mechanism makes CBBA robust to task and environment changes. Note that the standard CBBA assumes synchronous communication and requires agents to broadcast their assignment intentions in each round, which may incur some communication overhead; later researchers have also proposed an asynchronous version (ACBBA) to reduce the frequency of communication and the network burden while maintaining convergence [33]. Overall, CBBA represents a mainstream approach among distributed task allocation algorithms, and we evaluate it as a decentralized benchmark; it obtains near-optimal task allocation schemes without the need for central control.

Auction 2-Opt Refinement Algorithm (Distributed Bundle/Bidding Algorithm, distributed) refers to Algorithm 4. The auction 2-opt refinement algorithm is used here to represent another class of distributed task allocation methods, used in contrast to CBBA. Specifically, the auction 2-opt refinement algorithm can be viewed as a distributed search or optimization algorithm [36,37]: agents communicate together to participate in a distributed algorithmic process that may employ mechanisms similar to bundle bidding, branch delimitation, etc., to decide on task allocation. As an example, one possible auction 2-opt refinement algorithm implementation is a distributed branch-and-bound algorithm, where each agent explores a portion of the task-allocated solution space separately, exchanging boundary information with each other for pruning searches in order to find a globally optimal or near-optimal solution in the centerless case. Another may be an improved bidding algorithm or a distributed auction variant such as a bidding scheme when grouping the tasks or adding complex constraints. Since, in the literature, the auction 2-opt refinement algorithm may refer to specific algorithms (e.g., the improved bundle algorithm proposed in one study), in this research, we use it as a representative of distributed algorithms in a broad sense [38]. We expect that auction 2-opt refinement algorithm-type algorithms can enhance the performance of distributed allocation, possibly by being closer to the optimum than CBBA in terms of solution quality or by having better adaptive behavior in dynamically changing scenarios. However, the price of this may be larger computational and communication overheads. Thus, by incorporating the auction 2-opt refinement algorithm, we can examine the trade-offs within distributed algorithms, e.g., near optimization vs. overhead. In our experiments, we evaluate the auction 2-opt refinement algorithm together with CBBA to cover different implementation styles among distributed methods.

Algorithm 3 CBBA for Distributed Task Assignment

Require: Agents $A = \{a_1, \dots, a_{n_A}\}$; Tasks $T = \{t_1, \dots, t_{n_T}\}$; a utility function

$$v : A \times T \rightarrow \mathbb{R}, \quad v(a_i, t_j) = \begin{cases} -\text{dist}(a_i, t_j), & \text{if } a_i \text{ is compatible with } t_j, \\ -\infty, & \text{otherwise,} \end{cases}$$

and a maximum iteration limit $K_{\max}$.

Ensure: A consistent assignment $\mathcal{A} \subset A \times T$.

1: **Step 1: Initialization.**

2: For each agent $a_i \in A$, define its ordered preference list

$$P(a_i) = (t_{i_1}, t_{i_2}, \dots, t_{i_{|T|}})$$

such that

$$v(a_i, t_{i_k}) \geq v(a_i, t_{i_{k+1}}), \quad k = 1, \dots, |T| - 1.$$

Initialize the assignment functions as:

$$S(a_i) \leftarrow \emptyset, \quad \forall a_i \in A, \quad T_A(t_j) \leftarrow \emptyset, \quad \forall t_j \in T.$$

Let $U \leftarrow A$ denote the set of unassigned agents.

3: **Step 2: Iterative Assignment Process.**

4: For $k = 1, \dots, K_{\max}$, perform:

1. **For each** $a_i \in U$:

- (a) If $P(a_i) = \emptyset$, then no further task is available for a_i ; continue to the next agent.
- (b) Otherwise, let t^* be the task at the head of $P(a_i)$ (i.e., the currently most preferred task by a_i).
- (c) **Case 1:** If t^* is unassigned (i.e., $T_A(t^*) = \emptyset$), then assign:

$$S(a_i) \leftarrow t^*, \quad T_A(t^*) \leftarrow a_i,$$

and remove a_i from U .

- (d) **Case 2:** If t^* is already assigned to an agent $a_{i'}$ (i.e., $T_A(t^*) = a_{i'}$), then compare the utilities:

$$v(a_i, t^*) \quad \text{vs.} \quad v(a_{i'}, t^*).$$

- If $v(a_i, t^*) > v(a_{i'}, t^*)$, reassign t^* to a_i by setting

$$S(a_i) \leftarrow t^*, \quad T_A(t^*) \leftarrow a_i,$$

and mark $a_{i'}$ as unassigned (i.e., $S(a_{i'}) \leftarrow \emptyset$) so that $a_{i'}$ is added to U .

- Otherwise, remove t^* from $P(a_i)$ (i.e., $P(a_i) \leftarrow P(a_i) \setminus \{t^*\}$) and continue with the next best task.

5: **Step 3: Termination.**

6: The iterative process terminates when either

$$U = \emptyset \quad (\text{all agents are assigned})$$

or after $K_{\max}$ iterations. In either case, the resulting assignment is given by

$$\mathcal{A} = \{(a_i, S(a_i)) \mid S(a_i) \neq \emptyset\}.$$

7: **return** $\mathcal{A}$.

Algorithm 4 Auction 2-Opt Refinement Algorithm for Distributed Task Assignment

Require: A set of agents $A = \{a_1, \dots, a_{n_A}\}$ and a set of tasks $T = \{t_1, \dots, t_{n_T}\}$. Assume a distance metric $\text{dist} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_+$ (e.g., Euclidean distance). A local search maximum iteration limit $K_{\max}$.

Ensure: A consistent assignment $\mathcal{A} \subset A \times T$ with reduced total cost.

1: **Step 1: Initial Assignment via Auction.**

2: Obtain an initial assignment $S_0 : A \rightarrow T$ (possibly partial) by employing an auction-based method. Define the induced assignment mapping as:

$$S_0(a_i) = t_j \quad \text{if agent } a_i \text{ is assigned task } t_j.$$

3: **Step 2: Define Total Cost.**

4: For a given assignment $S : A \rightarrow T$, define the total cost

$$C(S) = \sum_{a_i \in A} \text{dist}(\ell(a_i), \ell(S(a_i))),$$

where $\ell(a_i)$ and $\ell(t_j)$ denote the locations of agent a_i and task t_j , respectively.

5: **Step 3: Local Exchange Optimization.**

6: Initialize the current assignment as $S \leftarrow S_0$ and set $k \leftarrow 0$.

7: **while** $k < K_{\max}$ **and** further improvement is possible **do**

8: **For all distinct pairs** $(a_i, a_j) \in A \times A$:

1. Denote the current tasks as $t = S(a_i)$ and $t' = S(a_j)$.

2. Compute the current cost for the pair:

$$C_{\text{current}} = \text{dist}(\ell(a_i), \ell(t)) + \text{dist}(\ell(a_j), \ell(t')).$$

3. Compute the potential cost if tasks are exchanged:

$$C_{\text{swap}} = \text{dist}(\ell(a_i), \ell(t')) + \text{dist}(\ell(a_j), \ell(t)).$$

4. **If** $C_{\text{swap}} < C_{\text{current}}$, **then** update the assignment:

$$S(a_i) \leftarrow t', \quad S(a_j) \leftarrow t,$$

and mark that an improvement has been achieved.

9: Increment $k \leftarrow k + 1$.

10: **end while**

11: **Step 4: Termination.**

12: Output the final assignment:

$$\mathcal{A} = \{ (a_i, S(a_i)) \mid S(a_i) \text{ is assigned} \}.$$

return $\mathcal{A}$.

2.3. RL-Based Approaches

RL Algorithm (Reinforcement Learning-Based Approaches, Centralized or Distributed). RL has been gradually applied to the field of multi-agent body task allocation in recent years, providing a data-driven solution pathway [39]. The RL algorithm allows an agent to learn a task allocation strategy by interacting with its environment: through continuous trial-and-error and feedback accumulation, the agent learns how to select tasks to optimize long-term returns. For single intelligences, task allocation can be modeled as a Markov decision process (MDP) and learned using algorithms such as deep Q-networks (DQNs) or policy gradients; for multi-intelligences, RL can be used with centrally trained, decentralized execution or fully decentralized multi-intelligence RL, with each agent acting as a learning intelligence. The strength of the RL approach is that it can adapt to complex

and dynamic scenarios, and, by learning to master patterns of environmental changes, the agent can exhibit near-optimal behavior in real-time decision-making [39]. Several studies have demonstrated that multi-agent deep RL allows agents to autonomously learn communication and collaboration strategies for task assignment [40,41]. For example, there is work that uses deep Q-learning to enable multiple agents to learn to exchange information about their respective observed tasks and coordinate their assignments, thus realizing distributed task assignment and achieving good performance in simulations [42]. RL algorithms are also used in dynamic task scenarios, such as unmanned fleets of vehicles receiving new task assignments in real time and dynamic assignments in urban environments through multi-agent body RL [43]. Since the inference of neural network policies is typically very fast, trained RL models can also decide on actions in near-constant time in large-scale systems, which means that RL methods have the potential to scale up to hundreds or even thousands of agents [44,45]. However, RL methods also have challenges: the training process can be time-consuming and requires the careful design of the reward function to guide the desired behavior. We include RL methods in the comparison to represent emerging agent allocation strategies, on the one hand, and to examine whether they can compete with traditional algorithms at different scales and dynamic environments on the other. However, we do not use them in the experiments because of their complexity.

2.4. Comparative Analysis and Gaps

Centralized vs. Distributed:

- Hungarian algorithms are centralized algorithms that need to centralize the information of all agents and tasks to achieve optimal matching and are suitable for smaller-scale scenarios.
- Bertsekas ϵ -auction algorithms, which can be implemented centrally or distributed, determine the task assignments through bidding and price adjustments among agents.
- CBBA and the auction 2-opt refinement algorithm are clearly distributed algorithms, where each agent communicates only with its neighbors, and are suitable for large-scale and dynamic environments, effectively avoiding centralized bottlenecks.
- RL methods, on the other hand, can be centrally trained and then decentralized and executed after to achieve distributed decision-making.

Optimality vs. Proximity:

- The Hungarian algorithm is guaranteed to find a globally optimal solution, but, in practice (especially in dynamic situations), it needs to be constantly recomputed.
- Bertsekas ϵ -auction algorithms and distributed methods (CBBA/auction 2-opt refinement algorithm) usually obtain approximate optimal solutions, but, due to their distributed architectures, they are more advantageous for large-scale problems.
- RL methods can make reasonable decisions with sufficient training, but they are not guaranteed to find optimal solutions. Furthermore, RL algorithms often face challenges such as training instability and convergence issues, the curse of dimensionality in high-dimensional state and action spaces, poor generalization to unseen scenarios, and a reliance on assumptions that rarely hold in practice, which limit their deployment in real-world multi-UAV systems [46].

Iteration and Convergence:

While the Hungarian algorithm generally requires only one iteration to find a match (without considering additional dynamics), the Bertsekas ϵ -auction algorithm, CBBA, and auction 2-opt refinement algorithm rely on multiple iterations and interagent communication (the ability to converge or reach a conflict-free allocation within a certain number of rounds), and the speed of convergence is an important evaluation metric. RL methods do

not require iterations in the inference phase (the model gives decisions directly), but the training process may require thousands of rounds of iterations to adjust the policy.

Computation and Communication Costs:

- Centralized Hungarian algorithms are computationally efficient at small scales, but both computational complexity and communication (the need to upload all messages to the central node) can be bottlenecks in large-scale problems.
- Distributed algorithms (auction, CBBA, auction 2-opt refinement algorithm) spread the computational load through local distribution and message exchange, but they therefore require more communication, which is especially costly in multiple iterations.
- RL methods reason very rapidly after training but carry greater computational costs in the training phase; moreover, whether they use distributed execution or not affects the communication overhead.

3. Algorithmic Framework

3.1. Hungarian Algorithm

Hungarian Algorithm (Centralized). The basic goal of the Hungarian algorithm is to solve the following optimization problem:

$$\min_{x_{ij}} \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (1)$$

where

- c_{ij} denotes the cost for agent i to perform task j (e.g., distance or time required);
- x_{ij} is the decision variable, which equals 1 if agent i is assigned to task j and 0 otherwise;
- The constraints ensure that each agent is assigned to exactly one task, and each task is assigned to exactly one agent.

This problem is known as the assignment problem, and the Hungarian algorithm guarantees finding a globally optimal solution in polynomial time [12,33].

3.2. Bertsekas ϵ -Auction Algorithm

Bertsekas ϵ -Auction Algorithm (Centralized/Distributed). Each agent bids for the task. We define the “utility” of each agent i for the task j as

$$u_{ij} = v_{ij} - p_j \quad (2)$$

where

- v_{ij} is agent i 's original evaluation of task j (usually proportional to the negative value of the cost, such as $v_{ij} = -c_{ij}$);
- p_j is the current price of task j .

The agent chooses the task that maximizes the utility, and each successful bid raises the task price by a small positive amount ϵ . The entire process iteratively updates the price until all agents have been assigned tasks [34,35].

3.3. CBBA

CBBA (Consensus-Based Bundle Algorithm, Distributed). CBBA's basic idea is as follows:

- For each agent i , precalculate the gain for each task j (usually expressed as a negative distance, such as $r_{ij} = -||position_i - position_j||$);
- Agents construct a set of task lists (bundles) sorted by gain, noted as B_i .

The agents then exchange their respective allocation intentions through local communication to solve the problem of interagent competition for the same task. Mathematically, it is difficult to write simply as a closed-form formula, but, at its core, it is desired that each agent's local decisions (greedy strategies) reach consistent convergence after communication negotiation, yielding a set of near-optimal allocations [33].

3.4. Auction 2-Opt Refinement Algorithm

Auction 2-Opt Refinement Algorithm (Distributed Bundle/Bidding Algorithm, Distributed). The auction 2-opt refinement algorithm can be viewed as an extension to CBBA that employs distributed search and local exchange mechanisms to improve the task assignment results, in addition to constructing bundles through greed [36,37]. For example, one possible auction 2-opt refinement algorithm implementation is the distributed branch delimitation algorithm; formally, we can describe it as

$$\min \sum_i c_{i,task(i)} \quad (3)$$

and satisfied by local exchange

$$c_{i,j'} + c_{k,j} < c_{i,j} + c_{k,j'} \quad (4)$$

This shows that the total cost of the system is reduced after the exchange.

3.5. RL Algorithm

RL Algorithm (Reinforcement Learning-Based Approaches, Centralized or Distributed). Instead of solving a mathematical planning model directly, the RL approach uses a network of policies (e.g., $\pi(a|s)$) to decide to “take action a in state s ”. The goal is to maximize the expected cumulative reward:

$$\max_{\pi} \mathbb{E} \left[\sum_{t=0}^T \gamma^t r_t \right] \quad (5)$$

where r_t can be designed as a negative reward associated with the cost of the task (e.g., a negative value for the distance traveled), as well as a task completion reward.

Although RL has shown significant progress in multi-agent decision-making and task allocation, it is not implemented in this study due to its high demands in terms of training data, interaction episodes, and policy convergence [40–42]. Compared to classical algorithms such as the auction, Hungarian, and CBBA methods, RL typically requires extensive simulation-based training to develop stable and effective policies, which considerably increases the implementation complexity and computational cost. Additionally, designing suitable state spaces, reward functions, and training procedures is a non-trivial task that lies beyond the primary scope of this study. As such, RL is considered a promising direction for future research but is not included in the current implementation.

4. Experiments

This section presents a comparative analysis of the four task allocation algorithms (Hungarian, auction, CBBA, auction 2-opt refinement algorithm) under three categories of experimental conditions (static tasks, dynamic tasks, and robustness tests under adverse conditions). Performance is evaluated using several metrics—total cost, completion rate, response delay, computational time, communication overhead, and fairness—with an emphasis on interpreting the numerical results to identify trends and trade-offs.

4.1. Experimental Settings

Our simulation framework, implemented in Python, provides a unified evaluation environment for various task allocation algorithms, including centralized optimization techniques such as the Hungarian algorithm and other custom plug-in modules. In this 2D simulator, agents and tasks are randomly distributed across a predefined area, with Euclidean distances used to compute travel costs, and tasks can appear either statically or dynamically. Each algorithm processes the current task and agent state information according to its unique strategy. Ultimately, the algorithms output agent–task assignments and record performance metrics such as the computation time and iteration count. This modular design not only standardizes the evaluation process across different algorithms but also facilitates direct and effective comparisons between them.

In detail, all experiments reported in this paper are conducted using Python (3.10). Numerical computations and data manipulations are performed using NumPy (1.24.2) and SciPy (1.10.1). Agent communications and network topologies are modeled using NetworkX (3.0). Reinforcement learning algorithms are implemented with Stable-Baselines3 (2.0.0), which relies on PyTorch (2.0.0) as the deep learning backend. Data visualization and plots are generated using Matplotlib (3.7.1). The experimental environment is set up on an Ubuntu 22.04 platform, equipped with a dual-socket configuration with Intel Xeon Gold 6338 processors.

4.2. Evaluation Metrics

- **Total Cost**
The total cost is the primary optimization metric, measured by the total travel distance or task completion time across all agents. This metric directly reflects efficient resource utilization, including fuel and time.
- **Completion Rate**
The completion rate represents the proportion of tasks that are successfully assigned and executed relative to the total generated. A high completion rate indicates that the algorithm can handle nearly all tasks, which is crucial in emergency scenarios or under task surges, preventing critical failures due to missed assignments.
- **Response Delay**
The response delay measures the time between a task being posted and its assignment to an agent. A low delay means that urgent tasks are addressed promptly. Centralized approaches may incur higher delays due to replanning the entire assignment, while distributed or RL methods tend to assign tasks immediately.
- **Computational Time**
The computational time quantifies the real time required to generate the assignment solution, typically measured in milliseconds or seconds. Shorter computational times support more frequent replanning and enhance system scalability, particularly as the problem size increases.
- **Communication Messages**
Communication messages quantifies the number and volume of messages exchanged in distributed algorithms, including bidding and consensus messages. A reduced overhead conserves bandwidth and energy and reveals how well different methods scale in terms of the messaging burden.
- **Fairness**
Fairness is defined based on the balance in task allocation among agents. Specifically, we quantify fairness by calculating the standard deviation of the number of tasks assigned to each agent. A smaller standard deviation indicates that tasks are more evenly distributed among agents, signifying higher fairness. Conversely, a larger

standard deviation reflects greater disparities, meaning that some agents bear significantly heavier workloads than others. This metric provides a straightforward measure of how equitably tasks are shared within the multi-agent system and is commonly employed in similar task allocation research.

4.3. Static Task Allocation

The objective of this experiment is to comparatively evaluate the performance of four task allocation algorithms in a static task allocation setting. In this configuration, all tasks are available and published at the initial time step ($t = 0$), thus creating a one-shot assignment scenario where the complete set of tasks is known a priori. This setup facilitates an isolated analysis of each algorithm's behavior and performance, eliminating the complications that arise from dynamic or time-varying environments.

The experimental scenarios are designed to encompass a range of problem complexities, ensuring that the conclusions drawn are robust and applicable across different scales of operation. Three distinct scenarios are implemented. The small-scale scenario involves 10 tasks to be assigned among 5 agents, a setting that serves as a baseline to reveal the fundamental operational differences among the algorithms. The medium-scale scenario increases the problem complexity by considering 50 tasks distributed over 15 agents, providing a more challenging environment that tests the scalability and efficiency of the proposed methods under moderately demanding conditions. The large-scale scenario presents an even more complex challenge, with 200 tasks allocated among 40 agents, thereby demonstrating the computational efficiency, convergence characteristics, and overall robustness of each algorithm.

To ensure statistical significance and account for any stochastic variability inherent in the allocation process, each scenario is independently repeated over 30 rounds. This repeated experimentation allows for a thorough assessment of the consistency and performance reliability across different task–agent configurations. All agents are assumed to have homogeneous communication and computation capabilities, and the experiments operate under the assumption of a static environment without any dynamic changes in task availability or agent functionality. This controlled experimental design is intended to provide clear insights into the relative merits and limitations of each algorithm in handling various scales of static task allocation problems.

4.4. Dynamic Task Allocation

Building on the insights gained from the static task allocation experiment, the next phase of our study introduces a more complex, dynamic environment in which tasks arrive over time. In this setting, the responsiveness and real-time processing capabilities of the allocation algorithms are tested as they contend with continuously emerging tasks. Rather than the one-shot scenario previously analyzed, this dynamic task flow scenario requires the system to continually update and reallocate tasks according to a predetermined schedule, thereby simulating realistic operational conditions.

The dynamic environment is designed around three distinct task publication frequencies to represent varying system loads and urgency levels. In the slow scenario, the system releases 2 tasks every 10 time units, creating relatively relaxed conditions that still capture the challenges of real-time processing. The medium scenario increases the complexity, with 5 tasks released every 5 time units, thus introducing a moderate level of temporal pressure and necessitating more efficient allocation processes. In the fast scenario, the highest system demand is simulated by releasing 10 tasks every 2 time units, placing significant strain on the algorithms to process and assign tasks rapidly and accurately.

The total simulation duration is set to 100 time units, ensuring that the dynamic flow of tasks is sustained over a sufficiently long period to reveal the operational nuances of each allocation strategy. To ensure that the results are statistically significant and robust, each dynamic scenario is repeated over 30 independent rounds. This comprehensive experimental setup enables a detailed assessment of how well the algorithms adapt to varying temporal complexities and provides deeper insights into their performance in environments where task arrivals are continuous and time-critical.

4.5. Robustness Testing

Following the analysis of both static and dynamic settings, we now extend our investigation to include robustness testing under abnormal conditions. This phase is specifically designed to challenge the resilience of the allocation algorithms when confronted with unexpected events, such as sudden task surges and agent failures, which are common in real-world scenarios.

In this experiment, the focus is on assessing how well the algorithms maintain their functionality and overall performance when facing disruptive events. The testing framework introduces three distinct abnormal conditions. First, in the surge scenario, a large number of tasks is suddenly released in the middle of the simulation, representing a scenario where the system experiences a sudden and significant increase in workload. Second, in the failure scenario, certain agents randomly experience failures during the operational period, rendering them temporarily unable to process or allocate tasks until they recover, thus simulating intermittent agent dysfunction. Lastly, the combined scenario simultaneously incorporates both the task surge and agent failures, thereby simulating a particularly challenging environment where multiple adverse events occur concurrently.

Each of these abnormal scenarios is simulated over a total time period of 100 time units, ensuring that the dynamic behavior of the system is thoroughly observed. To account for stochastic variations and to reinforce the statistical robustness of the findings, each scenario is independently repeated over 20 rounds. This rigorous experimental design provides a comprehensive evaluation of the algorithms' robustness under conditions of unpredictability and stress, offering valuable insights into their practical resilience and reliability in the face of operational anomalies.

The robustness tests introduce unreliable communications and robot failures. Under message loss, distributed algorithms (auction, auction 2-opt refinement algorithm, CBBA) show degradation: CBBA is the most sensitive due to its multi-round consensus, leading to delayed or missed assignments; the auction and auction 2-opt refinement algorithms recover via timeout and rebidding mechanisms, albeit with suboptimal allocations. The centralized Hungarian method remains robust provided that robots can communicate with the center, but network partitions remove isolated robots from the pool.

In robot failure scenarios, all methods experience proportional declines in their completion rates. The auction and auction 2-opt refinement algorithms rebid the tasks of failed robots promptly, incurring higher costs and delays for reassignment. CBBA requires additional consensus iterations to redistribute tasks, introducing longer interruptions. The Hungarian controller recomputes optimal assignments upon failure, minimizing cost increases but pausing execution during replanning.

5. Results

The following subsections detail the results for each scenario type and explain the observed behavior of the algorithms with respect to each evaluation metric.

In all experiments, the total cost is measured in meters (m); the completion rate is expressed as a percentage (%); the response delay and computational time are both

measured in seconds (s); communication messages is reported as a count of messages; and fairness is measured as a dimensionless number (standard deviation).

5.1. Static Task Allocation

Tables 1–3 present the comparative results of the four task allocation algorithms under three different static scenarios with increasing problem sizes. The following analysis and discussion are based on the data presented in these three tables, providing a systematic comparison of the algorithms' effectiveness and efficiency as the problem size increases.

Table 1. The table shows the performance of four task allocation algorithms in a small and static task allocation setting (10 tasks to be assigned among 5 agents).

Method	Total Cost	Completion Rate	Response Delay	Computational Time	Communication Messages	Fairness
Hungarian	92.2 ± 26	0.5 ± 0	0 ± 0	$(212.5 \pm 32.7) \times 10^{-6}$	0 ± 0	0 ± 0
Auction	103.5 ± 32.1	0.5 ± 0	0 ± 0	$(456.1 \pm 84.9) \times 10^{-6}$	6.17 ± 2.04	0 ± 0
CBBA	103.6 ± 32.5	0.5 ± 0	0 ± 0	$(214.2 \pm 19.3) \times 10^{-6}$	7.2 ± 1.73	0 ± 0
Auction 2-opt refinement algorithm	98.1 ± 27.7	0.5 ± 0	0 ± 0	$(628.5 \pm 92.3) \times 10^{-6}$	10 ± 0	0 ± 0

Table 2. The table shows the performance of four task allocation algorithms in a medium and static task allocation setting (50 tasks to be assigned among 15 agents).

Method	Total Cost	Completion Rate	Response Delay	Computational Time	Communication Messages	Fairness
Hungarian	121.7 ± 21.6	0.3 ± 0	0 ± 0	$(2783 \pm 250) \times 10^{-6}$	0 ± 0	0 ± 0
Auction	115.9 ± 13.5	0.3 ± 0	0 ± 0	$(3199 \pm 221) \times 10^{-6}$	17.4 ± 3.14	0 ± 0
CBBA	120.7 ± 18.5	0.3 ± 0	0 ± 0	$(2804.4 \pm 37.2) \times 10^{-6}$	19 ± 2.77	0 ± 0
Auction 2-opt refinement algorithm	122.8 ± 20.9	0.3 ± 0	0 ± 0	$(4972 \pm 334) \times 10^{-6}$	105 ± 0	0 ± 0

Table 3. The table shows the performance of four task allocation algorithms in a large and static task allocation setting (200 tasks to be assigned among 40 agents).

Method	Total Cost	Completion Rate	Response Delay	Computational Time	Communication Messages	Fairness
Hungarian	150.8 ± 12.4	0.2 ± 0	0 ± 0	$(28,574 \pm 511) \times 10^{-6}$	0 ± 0	0 ± 0
Auction	150.4 ± 13.5	0.2 ± 0	0 ± 0	$(28,870 \pm 1020) \times 10^{-6}$	45.2 ± 6.51	0 ± 0
CBBA	152.7 ± 11.7	0.2 ± 0	0 ± 0	$(29,505 \pm 424) \times 10^{-6}$	45.8 ± 2.54	0 ± 0
Auction 2-opt refinement algorithm	152.1 ± 10.8	0.2 ± 0	0 ± 0	$(42,380 \pm 5570) \times 10^{-6}$	780 ± 0	0 ± 0

Under the conditions described above, the centralized Hungarian algorithm produced the lowest-total-cost solution among the four methods, as expected given its optimal assignment guarantee. The auction-based method and the consensus-based algorithms (CBBA and its distributed variant, the auction 2-opt refinement algorithm) achieved total costs that were only marginally higher than that of the Hungarian baseline (typically within a few percent of the minimum), reflecting their effectiveness in approximating the optimal allocation through iterative bidding and negotiation.

In our static assignment context, the completion rate is defined as the ratio of tasks assigned in a single, one-shot allocation to the total number of tasks. Since each agent can only execute one task at a time, when the number of tasks exceeds the number of agents (which is common in static benchmarks), only a subset of tasks can be matched and scheduled in this single-step assignment. Therefore, the completion rate in this context reflects the one-time matching efficiency of each algorithm and is typically less than 100% whenever the number of tasks exceeds the number of agents. In contrast, dynamic task assignment experiments allow agents to iteratively take on new tasks as they become available, eventually achieving full task completion over time.

The initial assignment response delay was minimal for most methods: the Hungarian method dispatched all tasks in a single batch after a brief computation (polynomial-time optimization completed in milliseconds for the tested problem sizes). The auction, CBBA, and auction 2-opt refinement algorithms incurred slightly longer assignment delays due to

multiple bidding or negotiation rounds; however, these delays remained in the order of seconds and did not significantly hinder execution.

Regarding computational effort, the Hungarian algorithm was the most time-efficient in the static scenario. The Hungarian algorithm's cubic complexity was manageable for the tested sizes. In contrast, the auction and auction 2-opt refinement algorithms required more computation time overall due to iterative bid evaluations, and CBBA added overhead from its two-phase bundle construction and conflict resolution process.

The communication overhead followed a similar pattern: the centralized Hungarian approach required only minimal messaging (gathering task information and broadcasting final assignments). The auction and auction 2-opt refinement algorithms exchanged multiple messages per task (bids, replies, and notifications), proportional to the number of bidding rounds, while CBBA incurred a high message count owing to its repeated consensus iterations. Fairness in task distribution was high and roughly equivalent across methods, as optimal or near-optimal assignments divided tasks nearly evenly among robots, resulting in a comparable workload balance.

5.2. Dynamic Task Allocation

Tables 4–6 summarize the performance of the four task allocation algorithms under dynamic task arrival scenarios of varying intensity. These tables provide the basis for the following comparative analysis, which examines how each algorithm responds to different levels of dynamism in task allocation and highlights their respective strengths and limitations under varying operational pressures.

Table 4. The table shows the performance of four task allocation algorithms in a small and dynamic task allocation setting (2 tasks to be released every 10 time units).

Method	Total Cost	Completion Rate	Response Delay	Computational Time	Communication Messages	Fairness
Hungarian	259.1 ± 41.4	1 ± 0	0 ± 0	$(110.1 \pm 3.96) \times 10^{-6}$	0 ± 0	0.999 ± 0.164
Auction	418.9 ± 59.9	1 ± 0	0 ± 0	$(3382.7 \pm 52.9) \times 10^{-6}$	$1 \times 10^3 \pm 0$	0.713 ± 0.162
CBBA	251.1 ± 50.1	1 ± 0	0 ± 0	$(134.47 \pm 6.29) \times 10^{-6}$	242.1 ± 11	0.967 ± 0.169
Auction 2-opt refinement algorithm	394.7 ± 64	1 ± 0	0 ± 0	$(3511 \pm 459) \times 10^{-6}$	10 ± 0	0.793 ± 0.138

Table 5. The table shows the performance of four task allocation algorithms in a medium and dynamic task allocation setting (5 tasks to be released every 5 time units).

Method	Total Cost	Completion Rate	Response Delay	Computational Time	Communication Messages	Fairness
Hungarian	1144.5 ± 45.4	0.8067 ± 0.0375	10.02 ± 1.51	$(113.7 \pm 16.7) \times 10^{-6}$	0 ± 0	1.568 ± 0.295
Auction	862.1 ± 87.2	0.4603 ± 0.0385	15.34 ± 1.98	$(472.3 \pm 60.4) \times 10^{-6}$	297.1 ± 46.9	0.96 ± 0.166
CBBA	890.9 ± 86.1	0.4753 ± 0.0397	12.32 ± 2.14	$(130.7 \pm 16.5) \times 10^{-6}$	149.4 ± 10.8	0.902 ± 0.193
Auction 2-opt refinement algorithm	911.5 ± 75.5	0.4687 ± 0.0289	16.23 ± 2.11	$(506.5 \pm 83.8) \times 10^{-6}$	37.63 ± 3.53	0.905 ± 0.191

Table 6. The table shows the performance of four task allocation algorithms in a large and dynamic task allocation setting (10 tasks to be released every 2 time units).

Method	Total Cost	Completion Rate	Response Delay	Computational Time	Communication Messages	Fairness
Hungarian	1141.1 ± 13.8	0.4213 ± 0.0186	20.31 ± 1.49	$(1750 \pm 277) \times 10^{-6}$	0 ± 0	2.285 ± 0.471
Auction	955.4 ± 89	0.09747 ± 0.00756	27.73 ± 2.22	$(1056 \pm 154) \times 10^{-6}$	140.7 ± 4.55	0.851 ± 0.159
CBBA	951.4 ± 70.7	0.09393 ± 0.00537	26.97 ± 2.11	$(860 \pm 76.6) \times 10^{-6}$	114.9 ± 13	0.789 ± 0.137
Auction 2-opt refinement algorithm	939.6 ± 97.4	0.0968 ± 0.00748	27.41 ± 2.36	$(1068.6 \pm 94.4) \times 10^{-6}$	64.67 ± 4.94	0.861 ± 0.235

In the dynamic task allocation scenario, tasks arrived over time, requiring continual assignment updates. The centralized Hungarian approach, rerun upon each task arrival, maintained a low total cost by recomputing near-optimal assignments for the evolving task set; however, this introduced noticeable delays for new tasks as the solver's workload grew. In contrast, the auction and auction 2-opt refinement algorithms handled tasks incrementally in a distributed manner: robots immediately bid on arriving tasks, yielding

prompt assignments and low response delays. Their cumulative total costs remained close to optimal, with a slight trade-off in optimality for responsiveness. CBBA adapted by integrating new tasks into each robot's bundle followed by consensus rounds, producing timely allocations at costs marginally above those of the auction-based methods.

The completion rates remained very high for all methods. Fairness in the dynamic context decreased for all methods relative to the static case: cost-driven algorithms tended to overutilize a subset of robots, while CBBA and the auction 2-opt refinement algorithm sometimes mitigated imbalances through consensus. Quantitatively, the variance in tasks per robot increased, with consensus-based algorithms showing a marginally better load balance than greedy approaches.

The communication overhead showed a clear scale-dependent trend. For small problem sizes, CBBA incurred the highest message volume due to its intensive all-to-all consensus. As the scale increased, however, the auction and auction 2-opt algorithms required substantially more messages because their iterative bidding processes grow rapidly with more agents and tasks. Centralized approaches kept the message counts consistently low. This demonstrates that, while CBBA is communication-intensive for small teams, it scales more efficiently than auction-based methods as the system size increases.

5.3. Robustness Testing

Figures 1–3 provide a visual comparison of the four task allocation algorithms across three distinct robustness testing scenarios, and Figure 4 illustrates the functionality and overall performance of the algorithms under a combined scenario involving multiple concurrent adverse events. These figures form the foundation for the following robustness analysis, enabling a nuanced assessment of each algorithm's adaptability and reliability under both normal operation and adverse events.

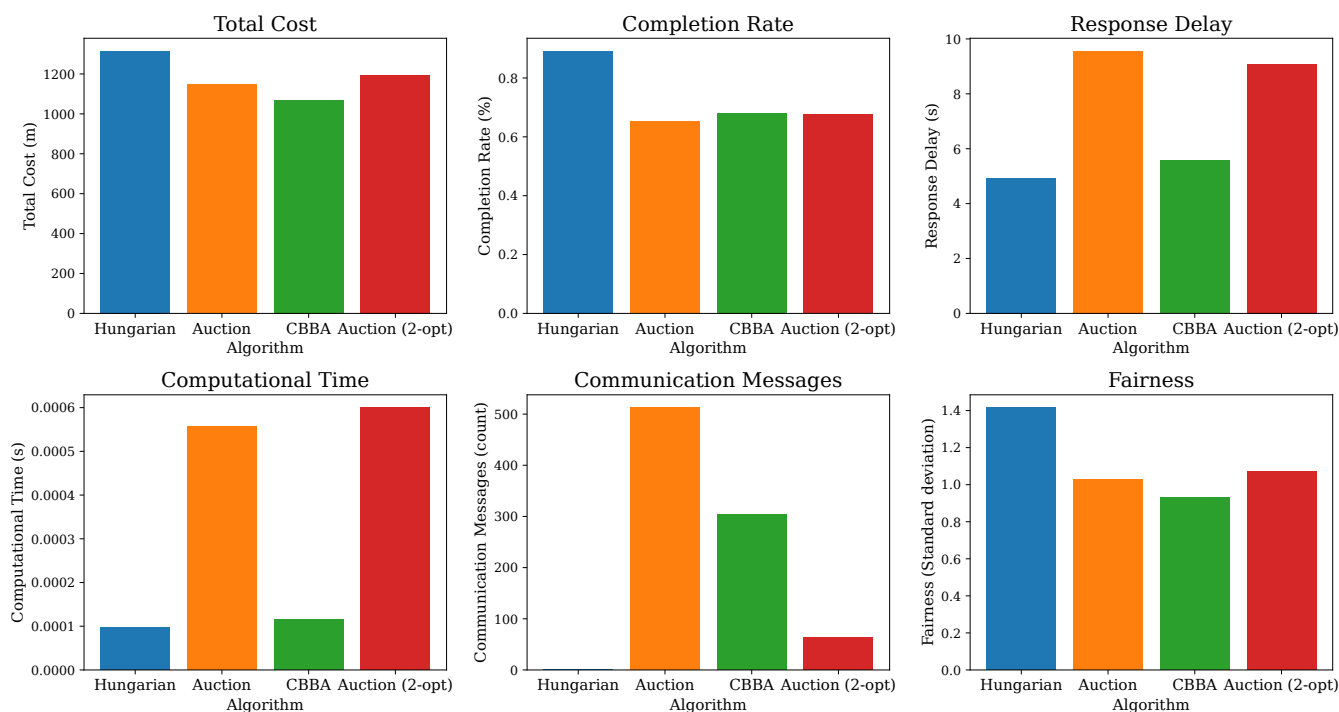


Figure 1. The figure shows the functionality and overall performance of four task allocation algorithms under normal scenario events (no abnormal conditions).

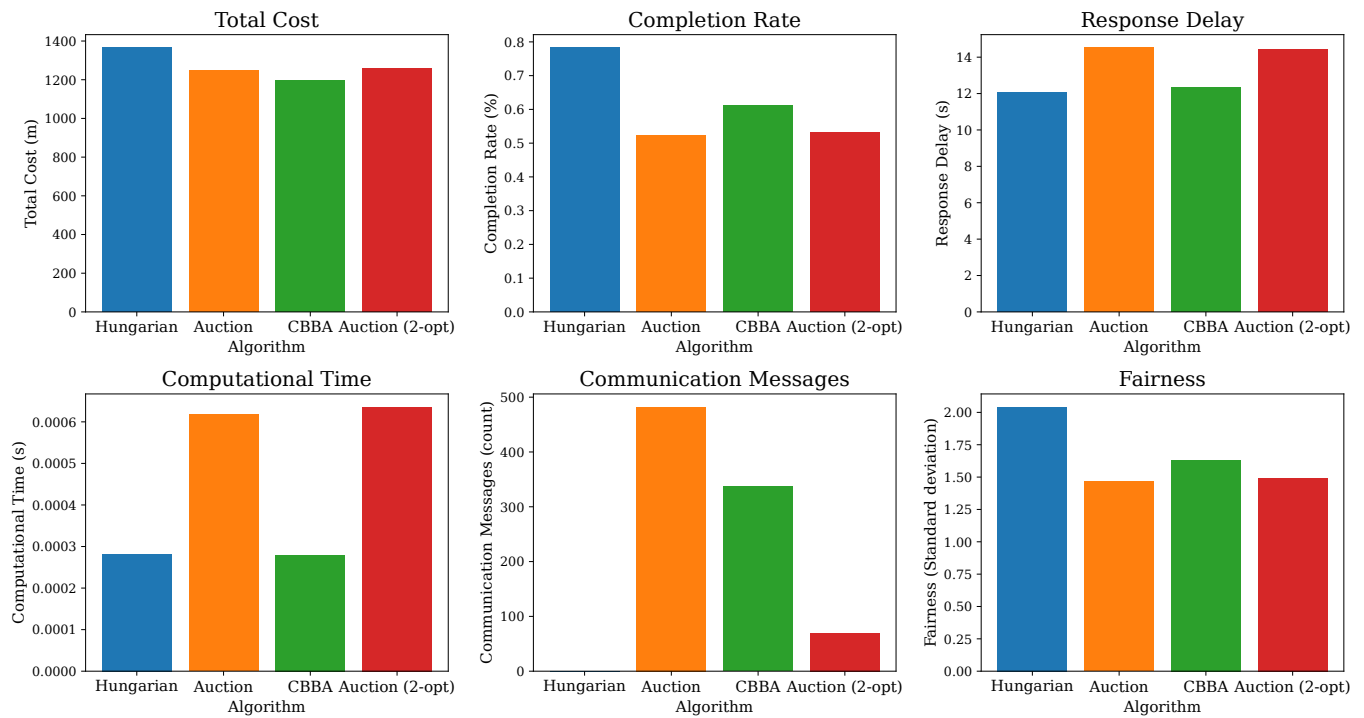


Figure 2. The figure shows the functionality and overall performance of four task allocation algorithms under surge scenario events (experiencing a sudden and significant increase in workload).

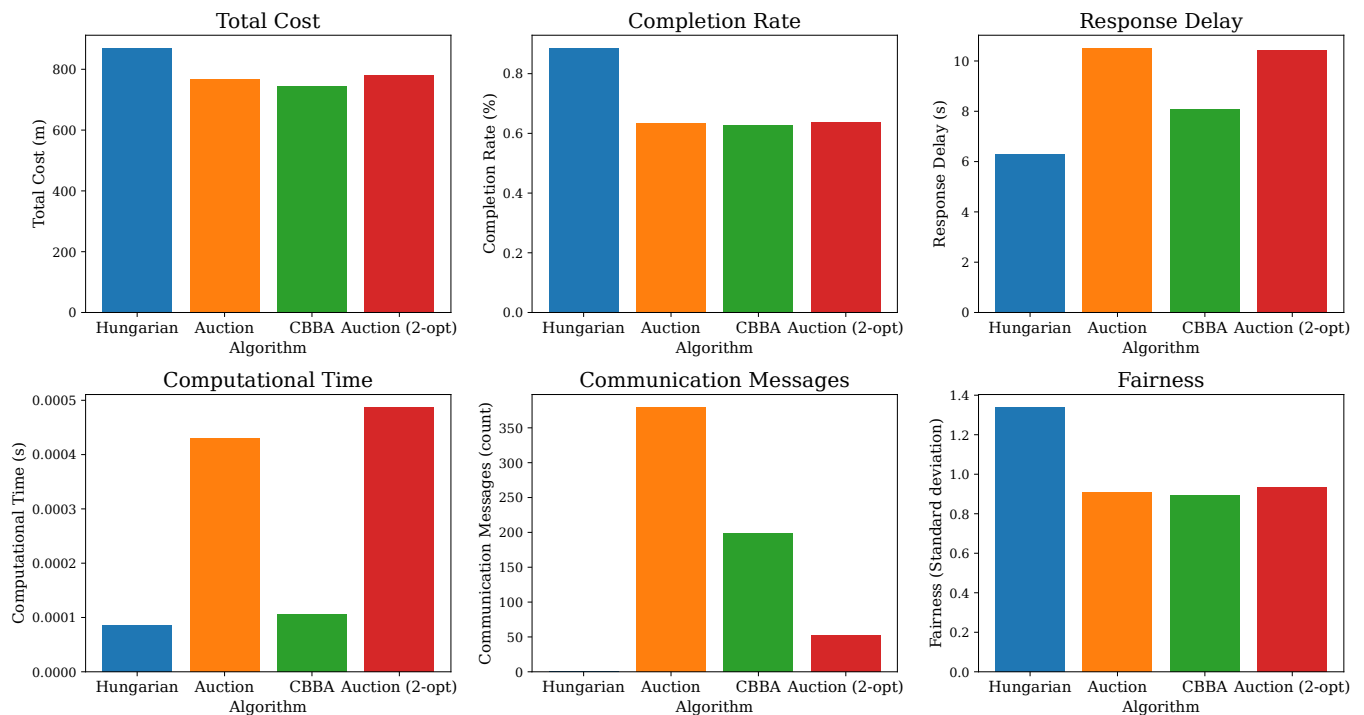


Figure 3. The figure shows the functionality and overall performance of four task allocation algorithms under failure scenario events (experiencing some agents temporarily unable to process or allocate tasks until they recover).

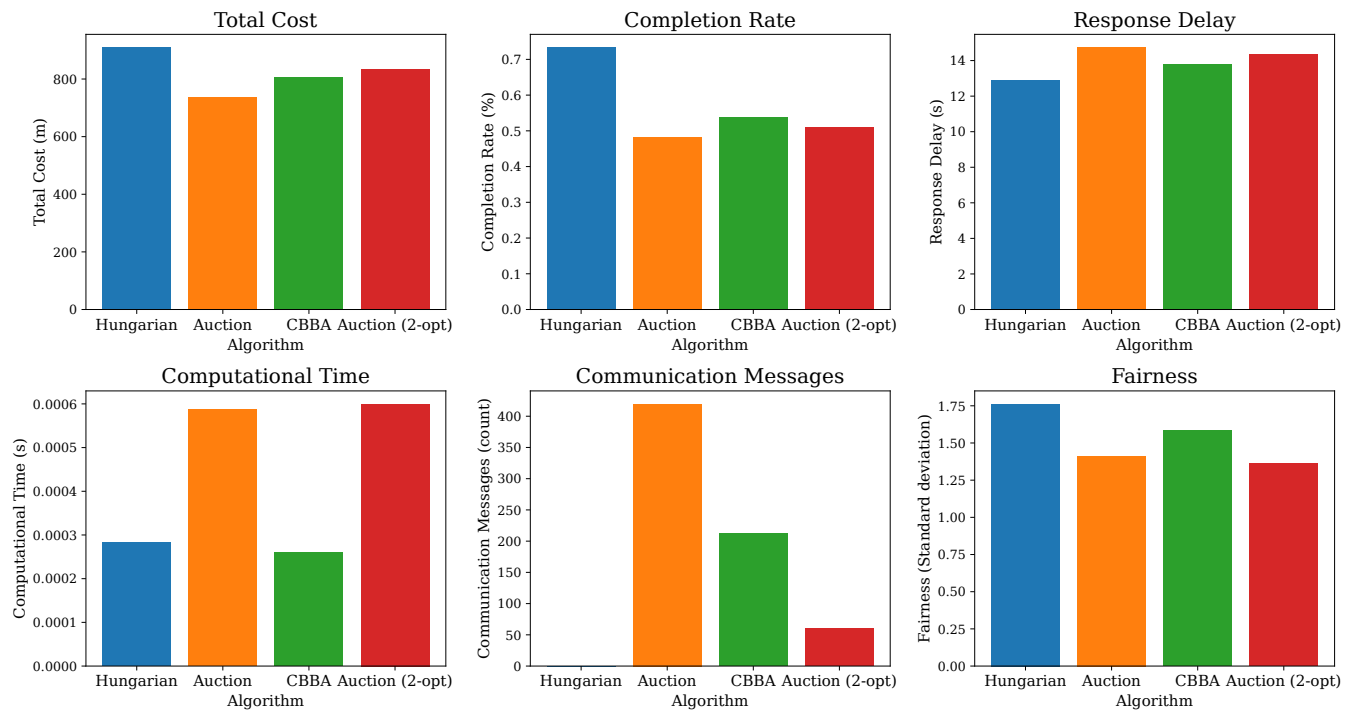


Figure 4. The figure shows the functionality and overall performance of four task allocation algorithms under combined scenario events (experiencing multiple adverse events that occur concurrently).

Adverse conditions increased the total cost across all methods, with the Hungarian algorithm best containing cost growth in failure scenarios and distributed algorithms incurring moderate penalties. The completion rates fell under severe network losses and failures, with CBBA being most impacted by message drops and all methods losing in-progress tasks upon failure. Response delays worsened under disturbances—distributed methods faced extra iterations or rebidding, and the Hungarian method paused during replanning. The communication overhead spiked for the auction and auction 2-opt refinement algorithms and CBBA under unreliable networks, while the Hungarian approach showed little change. The fairness decreased as failures and partitions concentrated the workload on surviving or connected robots.

5.4. Limitations and Problems in the Experiments

In our experiments, there were several problems that existed. The Hungarian algorithm was originally designed to solve square ($n \times n$) assignment problems. When the numbers of agents and tasks are not equal ($n \neq m$), as in our experiments, the algorithm exhibits several limitations. To handle this, the cost matrix must be padded with dummy rows or columns to ensure that it is a square. However, this introduces artificial assignments that do not have real-world meaning. Thus, this algorithm can usually obtain the optimal solution when allocating tasks from n to n , but it may not be the best allocation scheme when allocating tasks from n to m .

Regarding the Bertsekas ϵ -auction algorithm, it can solve both the n -to- n assignment problem and the n -to- m problem. However, it can only yield an approximately optimal solution rather than a globally optimal one. The Bertsekas ϵ -auction algorithm is essentially a discrete optimization method based on price adjustments. Since task prices are updated in fixed increments of ϵ , the algorithm can only achieve an approximately optimal solution with an error bounded by $n \cdot \epsilon$. When the utility values are real numbers, this discrete update mechanism may obscure subtle differences between assignments, preventing the algorithm from precisely identifying the true optimum. Therefore, without employing

ϵ -scaling, the Bertsekas ϵ -auction algorithm cannot guarantee global optimality. The auction 2-opt refinement algorithm, as a variant of the normal ϵ -auction algorithm, has the same limitation.

Regarding the CBBA algorithm, although it offers strong flexibility and distributed capabilities for n-to-m task allocation problems, it also exhibits several limitations under such asymmetric configurations. When the number of agents significantly exceeds the number of tasks, the algorithm tends to produce unbalanced assignments, where a few agents monopolize most tasks, while others remain idle. Conversely, in scenarios with a high task-to-agent ratio, the communication overhead increases substantially and frequent bundle reconstructions due to conflicts reduce the convergence efficiency. Furthermore, CBBA's greedy local bundle construction lacks global path optimality and offers limited control over the number of tasks assigned per agent.

5.5. Experimental Summary

Overall, each algorithm upheld its core strengths and weaknesses under adversity: the Hungarian method retained cost-optimality but lacked agility; the auction, CBBA, and auction 2-opt refinement algorithms adapted flexibly at the expense of communication and costs under stress. No single approach is universally superior; the choice depends on which performance criteria—cost, responsiveness, fault tolerance, or communication efficiency—are most critical for the intended operational context.

6. Conclusions

This work has presented a unified evaluation framework for the comparison of four multi-UAV task allocation algorithms, namely the centralized Hungarian algorithm, the Bertsekas ϵ -auction algorithm, the consensus-based bundle algorithm, and the auction 2-opt refinement algorithm, under realistic drone mission constraints such as limited flight time, propulsion energy endurance, and intermittent communication. By embedding UAV-specific cost models (e.g., energy consumption per flight segment) and modeling communication topologies reflective of line-of-sight restrictions, our simulations captured the key operational challenges faced by drone swarms in applications like area surveillance, search and rescue, and cooperative inspection.

In static mission scenarios, where all tasks are known a priori, the centralized Hungarian method consistently achieves the lowest total travel distance and energy usage, making it the natural choice for small-scale aerial mapping or preplanned inspection paths. However, as the number of UAVs and tasks grows or when new waypoints emerge mid-flight, the need to replan globally introduces non-negligible delays and imposes heavy communication burdens on the ground station or lead drone.

Auction-based heuristics and the auction 2-opt refinement algorithm strike a balance between optimality and responsiveness by allowing each drone to bid locally for incoming tasks. These methods deliver near-optimal allocations with only moderate computational overheads, enabling drones to react quickly when, for example, a sudden hotspot appears in a wildfire monitoring mission. Their decentralized nature also reduces the reliance on a single point of control, albeit at the cost of extra message exchanges to maintain consistent task prices.

The CBBA approach further enhances mission robustness by employing a consensus mechanism: when a drone fails or its propulsion energy runs critically low, the remaining UAVs renegotiate task bundles through peer-to-peer communication, redistributing responsibilities without central intervention. This resilience comes with increased messaging, making CBBA most suitable for high-stakes operations like disaster response, where the need for reliability outweighs the need for bandwidth conservation.

No single algorithm universally outperforms the others across all UAV mission profiles. Instead, practitioners should select based on the mission's tolerance for cost deviations, the permissible latency, the communication environment, and the failure risk. For tightly choreographed aerial inspections in urban canyons with reliable links, a centralized solver might suffice. In contrast, rapidly evolving search-and-rescue scenarios over rugged terrain require auction-based or consensus-driven methods.

Looking ahead, we plan to explore hybrid architectures that combine the precise cost minimization of centralized planners with the adaptivity of decentralized, learning-based policies. Embedding advanced fairness and load-balancing objectives will help to prevent individual drones from exhausting their batteries prematurely. We will also optimize consensus protocols for bandwidth-limited radio networks and validate our findings on physical UAV platforms, accounting for real-world energy profiles, heterogeneous payloads, and environmental disturbances such as wind gusts.

By strategically integrating the strengths of diverse coordination paradigms like global optimality, rapid replanning, and fault-tolerant consensus, we move closer to achieving autonomous drone swarms that can safely, efficiently, and reliably execute complex missions in the dynamic airspace of tomorrow.

Author Contributions: Conceptualization, S.Z. and S.S.; methodology, S.Z., Z.M., N.C. and Y.S.; software, Z.M., N.C. and Y.S.; validation, Z.M., N.C. and Y.S.; formal analysis, S.Z., Z.M., N.C. and Y.S.; investigation, S.Z., Z.M., N.C. and Y.S.; writing—original draft preparation, S.Z., Z.M., N.C. and Y.S.; writing—review and editing, S.Z., Z.M., N.C. and S.S.; project administration and funding, S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Source code and simulation data for the multi-UAV task allocation algorithms are publicly available at <https://github.com/YunzeSong/Multi-UAVs-Task-Allocation-Algorithms.git> (accessed 18 April 2025). For assistance, please contact the corresponding author.

Conflicts of Interest: The authors declare conflict of interest.

References

1. Lu, X.; Song, H.; Ma, H.; Zhu, J. A Task-Driven Multi-UAV Coalition Formation Mechanism. *arXiv* **2024**, arXiv:2403.05108. [CrossRef]
2. Zhang, C.; Xu, C.; Li, G.; He, B. A distributed task allocation approach for multi-UAV persistent monitoring in dynamic environments. *Sci. Rep.* **2025**, *15*, 6437. [CrossRef] [PubMed]
3. Zhou, X.; Wen, X.; Wang, Z.; Gao, Y.; Li, H.; Wang, Q.; Yang, T.; Lu, H.; Cao, Y.; Xu, C.; et al. Swarm of micro flying robots in the wild. *Sci. Robot.* **2022**, *7*, eabm5954. [CrossRef]
4. Liu, J.; Li, F.; Jin, X.; Tang, Y. Distributed Task Allocation for Multi-Agent Systems: A Submodular Optimization Approach. *arXiv* **2024**, arXiv:2412.02146. [CrossRef]
5. Wang, C.; Mei, D.; Wang, Y.; Yu, X.; Sun, W.; Wang, D.; Chen, J. Task allocation for Multi-AUV system: A review. *Ocean Eng.* **2022**, *266*, 112911. [CrossRef]
6. Hu, J.; Bruno, A.; Zagieboylo, D.; Zhao, M.; Ritchken, B.; Jackson, B.; Chae, J.Y.; Mertel, F.; Espinosa, M.; Delimitrou, C. To Centralize or Not to Centralize: A Tale of Swarm Coordination. *arXiv* **2018**, arXiv:1805.01786. [CrossRef]
7. Qamar, R.A.; Sarfraz, M.; Rahman, A.; Ghauri, S.A. Multi-criterion multi-UAV task allocation under dynamic conditions. *J. King Saud Univ.-Comput. Inf. Sci.* **2023**, *35*, 101734. [CrossRef]
8. Lemaire, T.; Alami, R.; Lacroix, S. A distributed tasks allocation scheme in multi-UAV context. In Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, 26 April–1 May 2004; Volume 4, pp. 3622–3627. [CrossRef]
9. Alqefari, S.; Menai, M.E.B. Multi-UAV Task Assignment in Dynamic Environments: Current Trends and Future Directions. *Drones* **2025**, *9*, 75. [CrossRef]
10. Yan, S.; Feng, J.; Pan, F. A Distributed Task Allocation Method for Multi-UAV Systems in Communication-Constrained Environments. *Drones* **2024**, *8*, 342. [CrossRef]

11. Alqefari, S.; Menai, M.E.B. A Hybrid Method to Solve the Multi-UAV Dynamic Task Assignment Problem. *Sensors* **2025**, *25*, 2502. [\[CrossRef\]](#)
12. Ghauri, S.A.; Sarfraz, M.; Qamar, R.A.; Sohail, M.F.; Khan, S.A. A Review of Multi-UAV Task Allocation Algorithms for a Search and Rescue Scenario. *J. Sens. Actuator Netw.* **2024**, *13*, 47. [\[CrossRef\]](#)
13. Ratnabala, L.; Fedoseev, A.; Peter, R.; Tsetserukou, D. MAGNNET: Multi-Agent Graph Neural Network-based Efficient Task Allocation for Autonomous Vehicles with Deep Reinforcement Learning. *arXiv* **2025**, arXiv:2502.02311. [\[CrossRef\]](#)
14. Durand, N.; Gianazza, D.; Gotteland, J.B.; Alliot, J.M. *Metaheuristics for Air Traffic Management*; John Wiley & Sons: Hoboken, NJ, USA, 2015. [\[CrossRef\]](#)
15. Skaltsis, G.; Shin, H.S.; Tsourdos, A. A Review of Task Allocation Methods for UAVs. *J. Intell. Robot. Syst.* **2023**, *109*, 76. [\[CrossRef\]](#)
16. Li, T.; Shin, H.S.; Tsourdos, A. Threshold Greedy Based Task Allocation for Multiple Robot Operations. *arXiv* **2019**, arXiv:1909.01239. [\[CrossRef\]](#)
17. Pawelek, A.; Lichota, P. Tactical and strategic air traffic sequencing with minimum-fuel trajectories. *J. Theor. Appl. Mech.* **2025**, *63*, 27–36. [\[CrossRef\]](#)
18. Liu, D.; Dou, L.; Zhang, R.; Zhang, X.; Zong, Q. Multi-Agent Reinforcement Learning-Based Coordinated Dynamic Task Allocation for Heterogenous UAVs. *IEEE Trans. Veh. Technol.* **2022**, *72*, 4372–4383. [\[CrossRef\]](#)
19. Choudhury, S.; Gupta, J.K.; Kochenderfer, M.J.; Sadigh, D.; Bohg, J. Dynamic Multi-Robot Task Allocation under Uncertainty and Temporal Constraints. *arXiv* **2020**, arXiv:2005.13109. [\[CrossRef\]](#)
20. Rinaldi, M.; Wang, S.; Geronel, R.S.; Primatesta, S. Application of Task Allocation Algorithms in Multi-UAV Intelligent Transportation Systems: A Critical Review. *Big Data Cogn. Comput.* **2024**, *8*, 177. [\[CrossRef\]](#)
21. Kurdi, H.A.; Aloboud, E.; Alalwan, M.; Alhassan, S.; Alotaibi, E.; Bautista, G.; How, J.P. Autonomous task allocation for multi-UAV systems based on the locust elastic behavior. *Appl. Soft Comput.* **2018**, *71*, 110–126. [\[CrossRef\]](#)
22. Ghassemi, P.; Chowdhury, S. Decentralized Task Allocation in Multi-Robot Systems via Bipartite Graph Matching Augmented with Fuzzy Clustering. *arXiv* **2018**, arXiv:1807.07957. [\[CrossRef\]](#)
23. Liu, R.; Seo, M.; Yan, B.; Tsourdos, A. Decentralized task allocation for multiple UAVs with task execution uncertainties. In Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 1–4 September 2020; pp. 271–278. [\[CrossRef\]](#)
24. Tereshchuk, V.; Stewart, J.; Bykov, N.; Pedigo, S.; Devasia, S.; Banerjee, A.G. An Efficient Scheduling Algorithm for Multi-Robot Task Allocation in Assembling Aircraft Structures. *arXiv* **2019**, arXiv:1902.08905. [\[CrossRef\]](#)
25. Sujit, P.B.; Beard, R. Distributed Sequential Auctions for Multiple UAV Task Allocation. In Proceedings of the 2007 American Control Conference, New York, NY, USA, 9–13 July 2007; pp. 3955–3960. [\[CrossRef\]](#)
26. Chen, Y.; Chen, R.; Huang, Y.; Xiong, Z.; Li, J. Distributed Task Allocation for Multiple UAVs Based on Swarm Benefit Optimization. *Drones* **2024**, *8*, 766. [\[CrossRef\]](#)
27. Xiao, K.; Lu, J.; Nie, Y.; Ma, L.; Wang, X.; Wang, G. A Benchmark for Multi-UAV Task Assignment of an Extended Team Orienteering Problem. *arXiv* **2020**, arXiv:2009.00363. [\[CrossRef\]](#)
28. Eser, M.; Yilmaz, A. Distributed Task Allocation for UAV Swarms with Limited Communication. *Sak. Univ. J. Comput. Inf. Sci.* **2024**, *7*, 187–202. [\[CrossRef\]](#)
29. Zhang, Z.; Jiang, J.; Xu, H.; Zhang, W.A. Distributed dynamic task allocation for unmanned aerial vehicle swarm systems: A networked evolutionary game-theoretic approach. *Chin. J. Aeronaut.* **2024**, *37*, 182–204. [\[CrossRef\]](#)
30. Raja, S.; Habibi, G.; How, J.P. Communication-Aware Consensus-Based Decentralized Task Allocation in Communication Constrained Environments. *IEEE Access* **2022**, *10*, 19753–19767. [\[CrossRef\]](#)
31. Yue, W.; Zhang, X.; Liu, Z. Distributed cooperative task allocation for heterogeneous UAV swarms under complex constraints. *Comput. Commun.* **2025**, *231*, 108043. [\[CrossRef\]](#)
32. Cao, M.; Nguyen, T.M.; Yuan, S.; Anastasiou, A.; Zacharia, A.; Papaioannou, S.; Kolios, P.; Panayiotou, C.G.; Polycarpou, M.M.; Xu, X.; et al. Cooperative Aerial Robot Inspection Challenge: A Benchmark for Heterogeneous Multi-UAV Planning and Lessons Learned. *arXiv* **2025**, arXiv:2501.06566. [\[CrossRef\]](#)
33. Johnson, L.; Ponda, S.; Choi, H.L.; How, J.P. A Consensus-Based Approach for Task Allocation in Dynamic Multi-UAV Teams. In Proceedings of the AIAA Guidance, Navigation and Control Conference, Chicago, IL, USA, 10–13 August 2009. [\[CrossRef\]](#)
34. Bertsekas, D.P. *Linear Network Optimization: Algorithms and Codes*; MIT Press: Cambridge, MA, USA, 1991.
35. Zavlanos, M.M.; Spesivtsev, L.; Pappas, G.J. A distributed auction algorithm for the assignment problem. In Proceedings of the 2008 47th IEEE Conference on Decision and Control, Cancun, Mexico, 9–11 December 2008; pp. 1212–1217. [\[CrossRef\]](#)
36. Mendoza, B.; Vidal, J.M. On bidding algorithms for a distributed combinatorial auction. *Multiagent Grid Syst.* **2011**, *7*, 101–120. [\[CrossRef\]](#)
37. Vidal, J.M. Bidding algorithms for a distributed combinatorial auction. In Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, Honolulu, HI, USA, 14–18 May 2007; ACM: New York, NY, USA, 2007; pp. 149–156.

38. Tovey, C.; Lagoudakis, M.G.; Jain, S.; Koenig, S. The Generation of Bidding Rules for Auction-Based Robot Coordination. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*; Parker, L.E., Schneider, F.E., Schultz, A.C., Eds.; Springer: Dordrecht, The Netherlands, 2005; pp. 3–14.
39. Fang, Z.; Ma, T.; Huang, J.; Niu, Z.; Yang, F. Efficient Task Allocation in Multi-Agent Systems Using Reinforcement Learning and Genetic Algorithm. *Appl. Sci.* **2025**, *15*, 1905. [[CrossRef](#)]
40. Foerster, J.N.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual Multi-Agent Policy Gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
41. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 6379–6390.
42. Zhang, K.; Yang, Z.; Başar, T. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In *Handbook of Reinforcement Learning and Control*; Springer: Cham, Switzerland, 2021; pp. 321–384. [[CrossRef](#)]
43. Qin, W.; Sun, Y.N.; Zhuang, Z.L.; Lu, Z.Y.; Zhou, Y.M. Multi-agent reinforcement learning-based dynamic task assignment for vehicles in urban transportation system. *Appl. Soft Comput.* **2021**, *111*, 107676. [[CrossRef](#)]
44. Yu, C.; Li, M.; Wang, Z.; Jiang, J.; Cai, Q.; Lu, Z. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *arXiv* **2021**, arXiv:2103.01955.
45. Papoudakis, G.; Christianos, F.; Albrecht, S.V.; Gross, R. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks, Online, 6–14 December 2021.
46. Dulac-Arnold, G.; Mankowitz, D.; Hester, T. Challenges of Real-World Reinforcement Learning. *arXiv* **2019**, arXiv:1904.12901. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.