Adaptive Market-Based Dynamic Task Allocation Under Environmental Uncertainty

Hasan Berke Ozturk[a], Nezih Bora Yavas[b] and Zafer Bingul[c]
Department of Mechatronics Engineering, Kocaeli University, Turkey

Keywords: Multi-Agent Systems, Uncertainty Theory, Swarm and Collective Intelligence, Decentralized Algorithms.

Abstract: This paper presents a novel consensus-based adaptive genetic-optimized auction (CAGA) algorithm to solve the dynamic task allocation (DTA) problem for a fleet of autonomous vehicles. The algorithm employs an auction routine for task assignment and a genetic algorithm (GA) to optimize task prices subject to the price update rule. The proposed algorithm is devised to achieve superior solutions in real-world applications. Hence, uncertainty theory was adopted to model uncertainties in task positions to create a realistic environment. In addition, Monte Carlo (MC) simulations are performed to effectively determine the degree of uncertainty. Several test scenarios have been carried out using other market-based methods, and the results illustrate the effectiveness of the algorithm.

1 INTRODUCTION

Autonomous vehicles (AUVs) have been extensively researched and used in numerous fields over the past decade due to their adaptable nature. Potential applications include mapping, reconnaissance, distributed sensor networks, hazardous metal handling, search and rescue operations, etc. Although their individual effectiveness has been proven, the deployment of multiple AUVs provides even more potent and appropriate solutions. Hence, the term multi-agent systems (MAS) has become a major topic as the significance of the problem continues to grow.

The primary objective of MAS is to ensure that agents in the fleet exhibit cooperative behaviors to effectively perform tasks. Also, from a task assignment perspective, it means multiple robots are tasked with achieving optimal assignments under certain constraints and maximizing their overall score. To establish a structure where a fleet of autonomous agents constantly perceives their surroundings and takes actions with regard to their benefit, coordination and cooperation among agents must be maintained. Several research studies have been carried out in this area, and the key points have been elaborated.

The centralized planners (Chen et al., 2024), (Hwang et al., 2022) assume a single computational unit that controls and coordinates each agent in the fleet by computing their cost and providing a communication network throughout a fixed location. Due to this centralized processing structure, the computational burden on agents is reduced, and they become simpler to build. However, coordinating agents from a stationary location restricts the workspace of the entire fleet, which further limits the potential tasks that the agents can handle and creates a single point of failure in the system.

Decentralized methods (Peng et al., 2024), (Zhang et al., 2024), (Ozturk et al., 2024b) have been proposed to solve the problems of centralized task allocation. This approach distributes the total computational load among agents rather than gathering it in a single unit. Thus, agents perform tasks based on their knowledge sets. Discarding the central structure that forces the system to fixate on a limited workspace has led to more robust and scalable agent cooperation. Conversely, to converge on an exact solution, agents should be able to share their status and information set among themselves through a particular network topology, which can result in intensive communication overhead and local optimality.

Market-based strategies are another well-known approach researched in the task allocation problem context (Ozturk et al., 2024a), (Wang et al., 2024). The auction algorithm mimics the auction environment, specifically by simulating bidders and auction-

[a] https://orcid.org/0009-0006-5730-4904
[b] https://orcid.org/0009-0008-9315-8494
[c] https://orcid.org/0000-0002-9777-9203

Ozturk, H. B., Yavas, N. B. and Bingul, Z.
Adaptive Market-Based Dynamic Task Allocation Under Environmental Uncertainty.
DOI: 10.5220/0013522600003970
In *Proceedings of the 15th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2025)*, pages 70-80
ISBN: 978-989-758-759-7; ISSN: 2184-2841

eers. First, bidders bid on the tasks they want to be assigned and calculate their maximum payoff. Afterward, auctioneers collect these bids and announce the highest bid as the contest winner.

Depending on how the problem is handled, market-based methods can be divided into single-item and combinatorial auctioning. Single-item auctions perform task-wise operations at each iteration. In contrast, combinatorial auctions present tasks as bundles (task sets), and agents bid on these bundles to minimize the path cost from their initial location. Researchers have developed variants of these approaches, including parallel single-item (PSI) and sequential single-item (SSI) auctions. Unlike single-item auction procedures, PSI performs auctions in a parallelized manner, in which agents are allocated to tasks via simultaneously performed auctions. Thus, it accelerates the assignment process but leads to suboptimal solutions. The SSI method, on the other hand, is a strategy that combines both combinatorial and PSI auctions to leverage their advantages. The method is based on a series of single-item auctions, assuming each agent is initially unallocated. To allocate tasks, agents place bids reflecting the increase in their smallest path cost that arises from winning the target they bid on. The agent offering the overall smallest bid is assigned to the corresponding target. Once agents determine the winner by observing the bids from the environment, the unassigned agents re-bid for the remaining tasks until all agents are assigned. Particularly in dynamic environments, SSI-based task updates are highly motivated since environmental quantities change drastically. Nonetheless, these auction methods neither provide a framework for resolving conflicting assignments nor guarantee optimal solutions in specific scenarios. Therefore, an agreement among the fleet should be consistently maintained to overcome these issues.

Consensus-based algorithms (Herty et al., 2024), (Bonandin and Herty, 2024) have thus gained prominence in addressing multi-agent coordination challenges. However, these approaches often face difficulties in achieving a common situational awareness (SA) among agents, that is, agreement that the perceived environment is the same for all of them. Although it is applicable across various network topologies, their implementation demands significant computational resources, and the convergence process can be notably time-intensive.

To resolve these problems, the consensus-based auction algorithm (CBAA) and, for multitask assignments, the consensus-based bundle algorithm (CBBA) have been introduced by (Choi et al., 2009). Both algorithms guarantee convergence on an agreed SA while ensuring conflict-free assignments. In contrast to traditional consensus approaches, these algorithms leverage a decentralized auction scheme in decision-making. Also, instead of agents' SA, they struggle to achieve an agreement on winning bid lists.

Unfortunately, all of the methods discussed above converge on a solution under the assumption of constant states and neglect uncertainty. This shortcoming implies they are unsuitable for real-world applications where the environment and its dynamics vary continuously.

This paper proposes a novel consensus-based adaptive genetic-optimized auction (CAGA) algorithm for dynamic task allocation of a multi-robot system. The algorithm can also consider the uncertainty in the environment and enable agents to make decisions based on the scenario characteristics. Therefore, the utilization of genetic algorithms (GA) is motivated by their ability to handle complex, multi-dimensional optimization problems where analytical solutions are infeasible to implement. In this context, the incremental constant (ε) serves as a critical parameter to regulate the pace of optimization, ensuring both convergence efficiency and computational feasibility.

The remainder of this paper is organized as follows: Section 2 investigates the related works proposed by other authors. Section 3 introduces the problem definition and preliminaries. Section 4 presents the proposed algorithm and its partitions. Section 5 illustrates the conducted simulations and provides test results, while Section 6 discusses the outcomes subject to the scenarios. Finally, Section 7 concludes the paper and gives valuable insights for future works.

2 RELATED WORKS

DTA has been investigated in detail, and various efforts have been made to solve this problem because of its importance. The proposed methods can be classified into two types: exact solutions and heuristics.

For heuristic methods, evolutionary-based approaches have mostly been utilized to solve the DTA. (Yan and Di, 2023) investigated the multi-robot task allocation problem and classified tasks as compulsory (must be completed) and functional (optional but beneficial). They aimed to optimize task assignments to minimize time costs by focusing on a novel hyper-heuristic algorithm. For this reason, researchers introduced and enlarged low-level heuristic (LLH) and high-level strategy (HLS) algorithms. LLH scores functional tasks based on an influence diffusion model, while HLS optimizes LLH parameters using a particle swarm optimization (PSO) al-

gorithm. The proposed algorithm was compared against classical greedy, metaheuristic, and SSI-based approaches. It was shown that the proposed algorithm is flexible and scalable. Nevertheless, it relies heavily on parameter tuning and struggles with extremely large or small robot groups, where simpler methods may suffice. Furthermore, adding one more layer increases the complexity of the implementation.

(Li et al., 2024) investigates task allocation challenges for heterogeneous unmanned aerial vehicles (UAVs), particularly in resource constraints and dynamic task demands. The authors proposed a heuristic allocation method grounded in the overlapping coalition formation game framework (OCF) to enhance resource utilization and task utility. This framework allows UAVs to operate under multiple conditions simultaneously, while the heuristic allocation method avoids repetitive and inefficient resource allocations. Moreover, the proposed task exit mechanism allows UAVs to exit coalitions when their utility contribution decreases. Although the algorithm demonstrates flexibility and dynamically responds to changes, the reliance on heuristic strategies results in high computational demands and a dependency on overly simplistic models, limiting its applicability in real-world scenarios.

In (Bischoff et al., 2024), re-optimization methods are introduced to tackle time-extended multi-robot task allocation problems involving task deletion and insertion in dynamic environments. The proposed optimization framework offers mechanisms to adapt to situations where new tasks are added or existing tasks are removed. As a heuristic approach, they present the cheapest maximum insertion cost heuristic (CMI) for task insertion by balancing complexity and performance. The introduced optimization heuristics are both scalable and efficient. This method establishes upper bounds on solution quality while ensuring predictability and reliability in dynamic systems. However, CMI heuristics can overestimate insertion costs, rendering the resulting bounds limited applicability and potentially inadequate in reflecting actual resource requirements.

Regarding exact solutions, these methods of DTA tend to achieve better convergence. One of the most commonly recognized strategies for solving DTA problems is the market-based approach, which operates mainly through auction routines. The study presented by (Hossain et al., 2023) offers a parallel task allocation framework using a multi-stage iterative combinatorial auction mechanism, introducing uncertainties in multi-robot environments. The paper aims to improve the task allocation process by exploiting different metrics, including task preferences, path collision avoidance, task prioritization, and execution deadlines. The proposed method offers two stages. In the initial stage, multiple rounds of bidding occur, and the temporary winner is determined. After that, a single round in which robots submit their final bids based on updated task values is executed for the final stage. The proposed mechanism is scalable and comprehensive, but it relies on a centralized entity, leading to a single point of failure in the system. Furthermore, the practical implementation of uncertain environments requires extensive fine-tuning.

3 PROBLEM DEFINITION

3.1 Task Allocation

In multi-robot task allocation (MRTA), there are m agents and n tasks, which can be defined as

$$\begin{aligned} \mathcal{A} &\triangleq \{a_1, a_2, \ldots, a_m\} \\ \mathcal{T} &\triangleq \{t_1, t_2, \ldots, t_n\} \end{aligned} \tag{1}$$

where $\mathcal{A}$ and $\mathcal{T}$ represent the sets of agents and tasks, respectively. The goal is for agents to find conflict-free assignments that maximize their payoff. Conflict-free means each agent is assigned only one task. The task assignment procedure continues until all agents are allocated to the tasks.

The overall score function for a conventional auction procedure is introduced. It can be expressed as

$$\max \sum_{i=1}^{m} \left(\sum_{j=1}^{n} (c_{ij} - p_{ij}) x_{ij} \right)$$

subject to

$$\begin{aligned} &\sum_{i=1}^{m} x_{ij} = 1, \quad \forall j \in \mathcal{T} \\ &\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in \mathcal{A} \\ &x_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{A} \times \mathcal{T} \end{aligned} \tag{2}$$

where c_{ij} is the benefit of agent a_i if it is assigned task t_j and p_{ij} is the price of task t_j for agent a_i. The x_{ij} is a binary decision variable, indicating that agent a_i is assigned task t_j if $x_{ij} = 1$ and 0 otherwise.

3.2 Auction Routine

Auction-based strategies are widely employed to solve task assignment problems. These routines promote competition among bidders, which significantly impacts the procedure's completion time.

Conventional auction schemes assume that agent a_i receives the maximum profit of c_{ij} for task t_j, where c_{ij} depicts the benefit of agent a_i if it bids for task t_j. It can be concluded that agent a_i will not bid for a task if it does not benefit from it.

Suppose that agent a_i pays the price p_j for task t_j; then, the payoff can be calculated as $c_{ij} - p_j$. Thereby, payoff functions of the agent and system can be attained in Eqs.(3) and (4) as follows:

$$\theta_i = \max_{t_j \in \mathcal{T}} (c_{ij} - p_j), \tag{3}$$

$$\Theta = \sum_{i=1}^{m} \theta_i \tag{4}$$

Here, $\mathcal{T}$ denotes the task set. In decentralized auction routines, if Eq.(3) is ensured for each agent in the fleet, then the agents are satisfied with their decision, and the global optimum is achieved. The system payoff Θ can be depicted as in Eq.(4).

If this is not the case, agents proceed to the next round, where the price update rule is executed.

An agent can raise the price offered by another agent to the extent that it becomes indifferent between its optimal and suboptimal payoffs. In light of this strategy, agents calculate their payoff for each task and bid on the task to maximize their benefit.

The optimal payoff is calculated as in Eq.(5). Where θ_i is the optimal payoff for the agent a_i and j^* is the task number corresponding to the optimal payoff.

$$\theta_i = g_{ij^*} = \max_{t_j \in \mathcal{T}} (c_{ij} - p_j) \quad t_{j^*} \in \mathcal{T}, \tag{5}$$

After that, calculate the suboptimal payoff $g_{ij_1^*}$ for agent a_i:

$$g_{ij_1^*} = \max_{t_j} (c_{ij} - p_j) \quad t_{j^*} \in \mathcal{T} \quad t_{j_1^*} \neq t_{j^*}, \tag{6}$$

where j_1^* is the task corresponding to the suboptimal payoff. By exploiting these equations, the next round offer of agent a_i can be introduced as

$$p_{ij^*} = d_{j^*} + g_{ij^*} - g_{ij_1^*} + \varepsilon \tag{7}$$

where d_{j^*} is the highest bid for task j^* offered in the previous round, and ε denotes the incremental constant, which significantly impacts the auction's course. In fact, it allows the system to converge on optimal solutions and overcome deadlocks (infinite iterations), whereas poor ε selection leads to higher computational resource consumption. Thus, it needs to be adjusted according to the specific scenario. Section 4 further elucidates how this value is optimized through a genetic algorithm (GA).

3.3 Communication Topology

The graph $G = (\mathcal{N}, \mathcal{V})$ is adopted to establish a network topology between decentralized agents. Namely, $\mathcal{N}$ is the set of nodes, and $\mathcal{V}$ is the set of vertices. If there is a direct communication link between two agents, that is, $(\delta, \eta) \subseteq \mathcal{V}$, then these two agents are adjacent. The number of agents in the network is $\mathcal{N}$, and the adjacency matrix C of G can be defined as in Eq.(8).

$$c_{\delta,\eta} = \begin{cases} 1, & \text{if } (\delta, \eta) \subseteq \mathcal{V} \\ 0, & \text{if } (\delta, \eta) \nsubseteq \mathcal{V} \end{cases} \tag{8}$$

$c_{\delta,\eta}$ is the element of the adjacency matrix C, a binary decision variable that denotes whether a link exists between two nodes δ and η. When dealing with uncertainties and executing the algorithm, it is assumed that the graph is undirected and fully connected.

3.4 Uncertainty Theory

When allocating tasks to agents, the severity of the uncertainty directly affects the decision-making process and, hence, how fast and effectively agents operate (ElGibreen and Youcef-Toumi, 2019).

Most conventional approaches do not consider uncertainty and its possible effects on the assignment procedure. Conversely, in cases where the environment is highly dynamic, task positions are not deterministic, and measurements vary over time. In this paper, uncertainty is introduced based on changes in the environment and is used to determine the accuracy of the agents' sensor measurements.

Uncertainty can be introduced with an uncertainty space (Liu and Liu, 2010), represented as $(\Gamma, \mathcal{L}, \mathcal{M})$ where Γ is a non-empty set, $\mathcal{L}$ is $\sigma-$algebra over a non-empty set Γ, and $\mathcal{M}\{\wedge\}$ is an uncertainty measure which indicates the level of belief that an event $\wedge$ will occur. This space has a measurable function defined as ξ that maps the quantities from uncertainty space to real numbers.

The lognormal distribution is implemented to model the uncertainty. Hence, $\xi \sim \mathcal{LOGN}(e, \sigma)$ and the uncertainty distribution is measured by $\mathcal{M}\{\xi \leq x\}$ for $x \geq 0$ in Eq.(9).

$$\Phi(x) = \left(1 + \exp\left(\frac{\pi(e - \ln(x))}{\sqrt{3}\sigma}\right)\right)^{-1} \tag{9}$$

On the other hand, finding the uncertainty distribution is not helpful since determining the unknown value of x is not possible in a dynamic environ-

ment. Thus, the distribution must be reconstructed using the uncertainty theory's operational properties. In this case, the inverse uncertainty distribution $\Phi^{-1}(\alpha)$ is obtained, and its uncertainty variable ξ has a regular distribution $\Phi(x)$, where $\alpha \in [0,1]$ is a confidence constant. For $\xi \sim \mathcal{LOGN}(e,\sigma)$, inverse uncertainty distribution can be derived as

$$\Phi^{-1}(\alpha) = \exp(e)\left(\frac{\alpha}{1-\alpha}\right)^{\sqrt{3}\sigma/\pi} \quad (10)$$

Utilizing the Eqs.(9) and (10), the overall score function can be formulated as

$$\max \sum_{i=1}^{m}\left(\sum_{j=1}^{n}\left(\left(1/\Phi^{-1}(\alpha)_{ij}\right) - p_{ij}\right)x_{ij}\right) \quad (11)$$

where $\Phi^{-1}(\alpha)$ is the inverse uncertainty distribution, and it represents the uncertainty of task positions. Eq.(11) is used when calculating payoffs.

4 METHODOLOGY

In this section, the proposed algorithm and its modules are introduced. Additionally, their working principles are elaborated.

4.1 Algorithm Framework

The proposed algorithm comprises two phases: First, a predefined number of CBAA operations are executed by agents **in parallel**, that is, each agent simultaneously performs a specific number of CBAA and aims to maximize its own score. The properties of locally executed CBAAs (level of possible task combinations, the amount of CBAA operations) are determined by GA's evolutionary parameters, including population size, crossover, and mutation. For each CBAA execution, agents randomly initialize different ε values ranging from $\left[5.10^{-6}, 5.10^{-4}\right]$ and find the best possible ε value by determining the overall maximum payoff (derived in Eq.(11)) among calculated CBAA procedures. Ultimately, the final CBAA is carried out with the best obtained ε values and allocates tasks to agents.

As a toy example of the proposed algorithm, assume that there are rooms where potential bidders are interested in the products being auctioned. The products are offered for sale to bidders in accordance with specific auction rules. Bidders submit bids on the most desirable item, thereby maintaining the momentum of the auction. The auctions are conducted in parallel sessions, with the same products being offered at varying levels of aggressiveness in each session. It is expected that the bidders maximize their profit, and if this criterion is not met, the auction extends for a predetermined number of days. At each auction, Bidders prioritize maximizing their own interests, and if the auction does not yield the desired profit, they must re-bid on a subsequent day. Moreover, prior to each auction, bidders conduct an internal assessment of the upcoming session, analyzing past results to refine their bidding strategies and optimize their offers accordingly. Subsequently, the most profitable auction combinations are identified, and buyers are encouraged to submit optimized bids for the corresponding products. Thus, by conducting multiple auctions, the profits of the bidders will be optimized, and the system will reach a state of equilibrium.

4.2 Monte Carlo Simulation

The Monte Carlo (MC) approach is adopted to determine the uncertain positions of the tasks and increase adaptability in a dynamic environment. Primarily, random distances between $(0,150)$ were calculated, and by using Eq.(9), several distance points were collected to determine the $\mathcal{LOGN}(e,\sigma)$ uncertainty pattern. Specifically, e and σ values are obtained over 1000 iterations, and the lognormal uncertainty distribution is acquired.

When locating tasks to place bids, task positions can be predicted by using the uncertainty theory. Uncertainty space can be reconstructed as $\{\mathcal{T}, D, \xi\}$, where $\mathcal{T}$ is the tasks, D is the distances, and ξ is the agent's benefit function. The goal is to find the uncertainty distribution of each task, in other words, $\xi_n \sim \mathcal{LOGN}(e,\sigma)$.

Subject to the task position ambiguity problem, suppose $\xi_1, \xi_2, \ldots, \xi_n$ are independent uncertain variables with a regular uncertainty distribution. $\Phi_1(x), \Phi_2(x), \ldots, \Phi_n(x)$ and the inverse uncertainty distribution for $\xi = f(\xi_1, \xi_2 \ldots \xi_n)$, where $f : \Re^n \rightarrow \Re$ is strictly increasing function denoted in Eq.(12). Thereby, agents have valuable insights and awareness regarding their benefits before assigning tasks in a dynamic environment.

$$\Psi^{-1}(\alpha) = f(\Phi_1^{-1}(\alpha), \Phi_2^{-1}(\alpha), \ldots, \Phi_n^{-1}(\alpha)) \quad (12)$$

4.3 CAGA Algorithm

The CAGA algorithm is proposed to solve the decentralized dynamic task allocation problem of the MAS. The algorithm is divided into two phases: the optimization phase and the assignment phase. Initially, with random ε combinations, agents conduct

parallel CBAA processes and compute their payoffs. Further, GA optimizes the whole procedure by adjusting the ε values.

input : agents $\mathcal{A}_i$, population size P
output: assignment matrix H

initialize y_t, generations P_g
for each agent a_i **do**
  initialize populations
end
for each generation **do**
  for each solution P **do**
    calculate fitness
    share fitness values
  end
  while *new population* $< P$ **do**
    select first parent from P
    select second parent from P
    crossover first and second parent
    mutate crossover-ed populations
    add new solutions to the P
  end
end
for each agent a_i **do**
  calculate global best solution
end
for each agent a_i **do**
  update parameters
end

Algorithm 1: Optimization Phase of CAGA Algorithm.

In the optimization phase, each agent initializes parameters, including population size, crossover, and mutation rate. Particularly, between the population members (each CBAA performed by each agent), combinations with the best payoffs are stored, and the remainder sizes are filled with the arithmetic crossover of former populations. In addition, by leveraging Gaussian distribution, populations exposed to crossover are mutated, and new epsilon combinations are obtained. After these operations, each combination is reserved, and agents proceed to the assignment phase.

Table 1: Symbol description of CAGA algorithm.

Symbol	Description
z_i	The task list of agent i
w_i	The winning bid list of agent i
v_i	The valid task list for agent i
q_{i,J_i}	The agent that currently assigned task J_i
$\mathbb{P}(\cdot)$	It is the indicator function that is unity if the argument is true and zero otherwise
G	The communication topology
k	Agent k is the neighbor of agent i

In the assignment phase, after determining the ε that provides the highest system payoff, agents carry out the original CBAA and assign themselves to tasks.

input : The agent set $\mathcal{A}$ the task set $\mathcal{T}$, the communication topology G and the initial price p_{ij}
output: system payoff

while *iteration* $<$ *max iteration* **do**
  for each agent a_i **do**
    $z_i(t) = z_i(t-1)$
    $w_i(t) = w_i(t-1)$
    if $\sum_j z_{ij} = 0$ **then**
      $v_{ij} = \mathbb{P}(c_{ij} > w_{ij}(t)), \forall j \in \mathcal{T}$
      if $v_i \neq 0$ **then**
        $J_i = \text{argmax}_j \, v_{ij} \cdot c_{ij}$
        $z_{i,J_i}(t) = 1$
        $w_{i,J_i}(t) = c_{i,J_i}$
        update the price p_{ij}
      end
    end
    send w_i to k with $g_{ik}(\tau) = 1$
    receive w_k from k with $g_{ik}(\tau) = 1$
    $w_{ij}(t) = \max_k g_{ik}(\tau) \cdot w_{kj}(t), \forall j \in \mathcal{T}$
    $q_{i,J_i} = \text{argmax}_k g_{ik}(\tau) \cdot w_{k,J_i}(t)$
    if $q_{i,J_i} \neq i$ **then**
      $z_{i,J_i}(t) = 0$
    end
  end
end

Algorithm 2: Assignment Phase of CAGA Algorithm.

In addition, the MC simulations help them effectively detect the exposed uncertainty and make decisions accordingly.

5 SIMULATION RESULTS

This section provides a detailed description of the crucial aspects of the simulation process and presents various test scenarios. In fact, small-, medium-, and large-scale applications are conducted, and the results are visualized to illustrate the adaptability of the CAGA algorithm.

5.1 Experimental Setup

The simulation setup introduces the preliminaries, including initial configurations, MC operations, and the evaluation metrics used for analysis. Initially, agents and task positions are set, and then agents execute the CAGA algorithm to determine assignments. Once targets are identified, the simulation begins to demonstrate scenarios at predefined scales. Due to environmental uncertainty, agents must periodically update their measurements for accurate task execution. Also, these updates should be synchronized to ensure feasible bid assessment and task re-allocation when necessary.

During simulations, the uncertainty degree is assumed to increase proportionally with distance and follow a lognormal distribution. The simulation environment adjusts the fleet's measurement frequency to improve assignment efficiency, considering the distance of the farthest agent from the task, as this agent experiences the highest level of uncertainty. In contrast, the time step is adjusted inversely to the degree of uncertainty, ensuring adaptive responsiveness. This relationship can be expressed as

$$\Delta t = \frac{\mathcal{C}}{max(\gamma) + \kappa} \tag{13}$$

where Δt represents time step, $\mathcal{C}$ signifies constant scale factor defined as 1, γ denotes the set of uncertainty degree ($\gamma \in [0,1]$) experienced by each agent and κ depicts the increment to avoid division by zero.

For the purposes of illustration, consider a scenario in which three agents (a_1, a_2, a_3) are involved in a given problem, with the agents positioned 10, 20, and 30 meters away from their designated tasks, respectively. According to the simulation, the agent positioned 30 meters away exhibits the highest degree of uncertainty. To determine the furthest one, agents share their degree of uncertainty with themselves. Later, they define the highest bid and use it to calculate Eq.(13) while updating their status. As the agents approach their assigned tasks, the uncertainty decreases, thereby enhancing the agents' confidence in their measurements and decreasing the measurement frequency. Conversely, in a scenario where the agents are distant from the tasks, the uncertainty is elevated, and the measurements are received at more frequent intervals.

The CAGA algorithm introduces an MC simulation to capture uncertainty characteristics at various levels of uncertainty. This way, agents will have more information about their workspace. Before performing assignments, agents are informed of the presented uncertainty ($\mathcal{LOGN}(3.6, 0.6)$) with the **historical data** collected from MC operations in a 100×100 workspace.

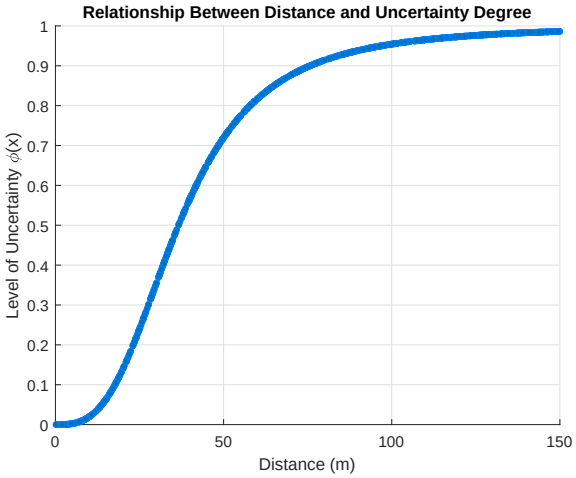

Figure 1: Lognormal uncertainty distribution function.

Figure 1 visualizes the uncertainty distribution adopted in this paper. By using this knowledge, agents have the capability to analyze the environment and make decisions more accurately.

```
Carry out MC simulations
while system progress < t do
    for current positions do
        calculate Euclidean distance D_ij
        if D_ij < 10^-6 then
            nullify the uncertainty
            coeff = 0
        end
        else
            calculate measured distance
            calculate measured target
        end
        calculate new positions
        calculate Δ_t value
    end
end
```

Algorithm 3: Simulation steps.

5.2 Results

Simulations for scales of 3×3, 5×5, and 10×10 were carried out, and the task assignments performed by the CAGA algorithm have been visualized. The proposed algorithm's performance metrics were collected using the same model created in Python.

For the simulations, the **initial** agent and task positions for small-, medium-, and large-scale applications were randomly defined within a 100×100

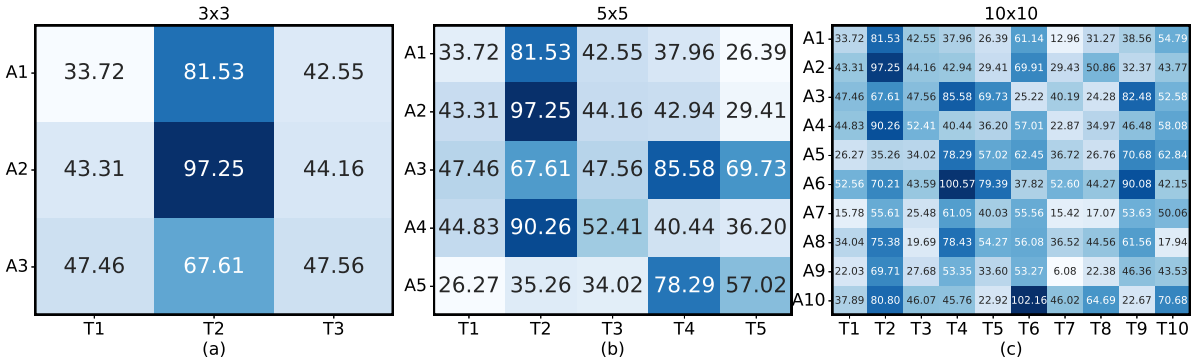

Figure 2: Initial payoff matrices for conducted scenarios.

workspace. The initial payoff matrices, which are shown in Figure 2, were derived by calculating the Euclidean distance between the agent-task pairs and

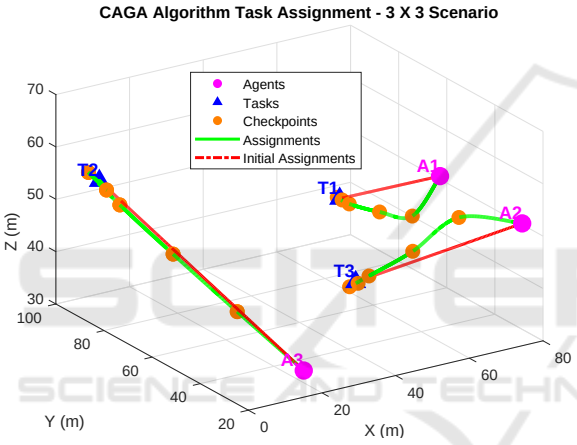

Figure 3: The CAGA algorithm performance for 3×3.

Table 2: Positions on XYZ (in meters) for 3×3 Scenario.

Agent Positions	Task Positions	Sensor Outputs
[54 23 65]	[47 54 54]	[32.1 72.2 40.51]
[75 21 53]	[52 55 37]	[38.5 76.1 41.4]
[17 23 34]	[2 85 57]	[42.2 58.6 42.8]

adding a $\mathcal{LOGN}(3.6, 0.6)$ uncertainty noise to these values. After that, at each checkpoint of the simulation, sensor outputs were obtained from the corresponding positions of the agents (To improve readability, only the first sensor measurement output is provided for each agent). These values are presented in Tables 2, 3, and 4, respectively.

The simulation mechanism devised for the scenarios is as follows: Agents and task locations are initialized for each procedure, and tasks are allocated by leveraging the CAGA algorithm subject to the payoff matrices in Figure 2. The simulation is divided into

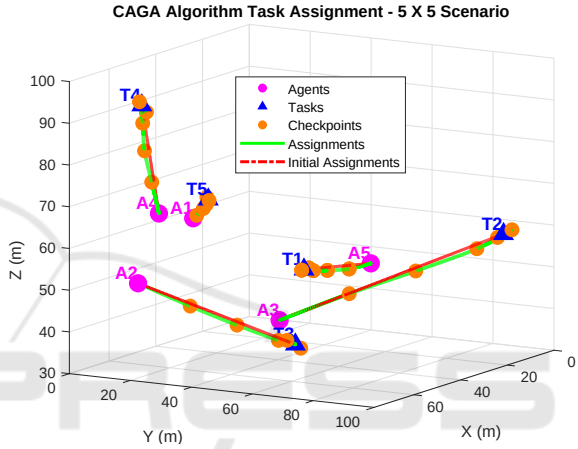

Figure 4: The CAGA algorithm performance for 5×5.

Table 3: Positions on XYZ (in meters) for 5×5 Scenario.

Agent Positions	Task Positions	Sensor Outputs
[54 23 65]	[71 41 74]	[70.3 40.2 73.6]
[75 21 53]	[52 55 37]	[53.6 52.6 38.1]
[17 23 34]	[2 85 57]	[4.27 75.6 53.5]
[53 11 65]	[76 23 96]	[74.3 22.1 93.8]
[21 56 51]	[47 54 54]	[45.8 54.9 53.8]

five time intervals, each containing a checkpoint. This structure allows agents to simultaneously update their measurements, refine their positions, and potentially reallocate tasks. At each checkpoint, they recalculate their payoffs to each task and, if a higher payoff exists, reallocate to another task by re-executing the CAGA algorithm. Since agents synchronously reach checkpoints and adjust their progression rates according to the agent most affected by the uncertainty, task conflicts will not arise.

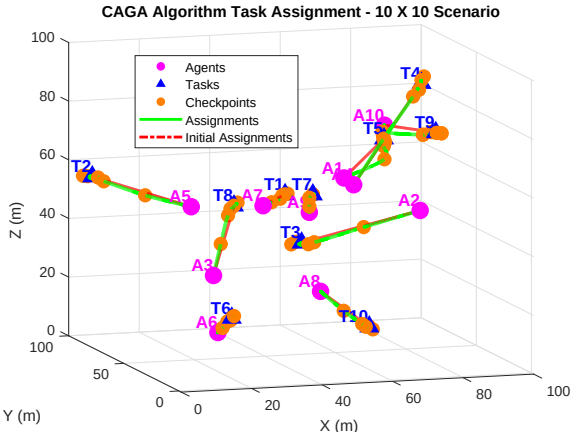

Figure 5: The CAGA algorithm performance for 10 × 10.

Table 4: Positions on XYZ (in meters) for 10 × 10 Scenario.

Agent Positions	Task Positions	Sensor Outputs
[54 23 65]	[71 41 74]	[84.2 40.9 74.4]
[75 21 53]	[52 55 37]	[54.2 51.6 38.5]
[17 23 34]	[25 30 56]	[24.6 29.6 55]
[53 11 65]	[76 23 96]	[74.6 22.2 94.1]
[21 56 51]	[2 85 57]	[3.69 82.4 56.4]
[22 35 12]	[15 0 24]	[15.4 1.98 23.3]
[37 42 53]	[47 54 54]	[46.7 53.7 53.9]
[53 41 23]	[64 32 12]	[63.5 32.3 12.4]
[47 32 52]	[48 32 58]	[47.9 32 57.8]
[78 63 74]	[86 42 75]	[71.2 41.8 74]

A small-scale application was initially conducted, with results visualized in Figure 3. Due to uncertainty and sensor errors, agents deviated from their trajectories. At each checkpoint, they reassessed their positions and, if necessary, reallocated tasks. Notably, at the second and third checkpoints, agents 2 and 3 switched tasks, but by the final checkpoint, updated payoffs led them to reallocate to their initial tasks. This behavior distinctly illustrates the proposed algorithm's resilience to uncertainty.

In medium-scale applications, as depicted in Figure 4, agents effectively navigated environmental uncertainty while maintaining their initial allocation paths, in contrast to the deviations observed in small-scale scenarios.

Ultimately, the CAGA algorithm was tested in a 10 × 10 scenario, with results presented in Figure 5 and Table 4. The simulation outcomes demonstrate that agents successfully maintained their initial task assignments while effectively adapting to uncertainties. Throughout the execution, they remained within their designated measurement corridors, ensuring stable and reliable task completion.

Interestingly, during the execution of the simulation scenarios, the uncertainty inherent in the environment does not significantly divert the agents from their trajectories. In contrast, results clearly show that the MC approach introduces adaptability to the CAGA algorithm, allowing agents to manage uncertainty effectively within dynamic environments.

6 DISCUSSION

The CAGA algorithm was compared with CBAA and other market-based approaches across different scenarios both in terms of computational time and total payoff. Particularly, the proposed optimal market-based (OMB) (Liu and Shell, 2013) and improved market-based (IDMB) (Trigui et al., 2014) approaches were performed, and results are shown in Table 5 and 6. All simulations were implemented in Python 3.12.7 and tested on a system with a 4-core 4.2 GHz i7-7700K CPU and 16 GB of memory.

Table 5: The payoff comparison of the algorithms. The boldface values are the best average for each algorithm.

Scenarios	Algorithms			
	CAGA	CBAA	OMB	IDMB
3 × 3	6.71	6.71	6.71	6.71
5 × 5	**13.82**	**13.82**	13.2	13.80
7 × 7	**29.13**	28.37	25.5	22.61
10 × 10	**50.59**	49.82	49.6	43.22
15 × 15	**75.689**	74.74	71.87	72.81
20 × 20	**125.86**	111.4	106.5	109.72
25 × 25	**182.32**	143.8	126.4	127.4
30 × 30	**209.09**	169.1	157.8	148.07
35 × 35	**248.06**	247.7	227.7	209.8
40 × 40	279.42	**281.6**	265.8	244.5

In the proposed algorithm, the assignments are derived from the last checkpoint of the simulations. On the other hand, since other algorithms do not utilize checkpoints, assignments are made directly based on the initial cost matrix. After that, cost matrices with uncertainties have been used to perform all assignments. Finally, cost matrices without uncertainty have been used to calculate payoffs to ensure a consistent evaluation of payoffs.

The payoff and runtime performances of the algorithms, as presented in Tables 5 and 6, were tested in 25 trials, and the arithmetic mean of the results was used for comparison. The results are calculated using

$100\,\lambda^{-1}$, where λ denotes the cost matrix. The matrix elements are derived based on the Euclidean distance between the positions of the agents and tasks. To prevent extremely small values and enhance the accuracy of the results, the inverse of the cost matrix is scaled by a factor of 100. According to the results, the proposed algorithm demonstrates superiority for small- and medium-scale applications regarding system payoff in an uncertain environment. It expands the search space and identifies possible solutions available to agents.

Table 6: The computational time comparison of the algorithms (in milliseconds). The boldface values are the best average for each algorithm.

Scenarios	Algorithms			
	CAGA	**CBAA**	**OMB**	**IDMB**
3×3	8.89	0.97	0.09	**0.03**
5×5	9.76	0.99	**0.17**	0.24
7×7	11.95	1.89	0.42	**0.30**
10×10	13.16	1.99	0.44	**0.38**
15×15	18.02	4.98	0.79	**0.71**
20×20	22.92	8.97	**0.86**	0.92
25×25	30.78	17.97	**1.15**	1.65
30×30	40.84	35.90	**2.06**	2.80
35×35	66.41	55.82	**3.36**	3.69
40×40	83.79	73.80	**3.84**	4.32

However, as the operational scale increases, the CAGA algorithm struggles with the growing number of uncertain conditions in the system, which ultimately leads to computational overhead. Table 6 illustrates the computational times of each algorithm across various applications. The results highlight the efficiency of the OMB algorithm, whereas the CAGA algorithm demonstrates relatively poor performance. Nevertheless, the CAGA algorithm accounts for uncertainty and can be easily adjusted for application scenarios where optimality precedes time sensitivity.

7 CONCLUSION

In this paper, the decentralized dynamic task allocation problem of MAS was introduced and solved with the proposed CAGA algorithm. The proposed algorithm was compared with several market-based methods and the well-known consensus-based decentralized task assignment algorithm CBAA, and the results are discussed. According to the simulations conducted in different operational sizes, it has been demonstrated that the proposed algorithm provides valuable outputs in terms of system payoff in an uncertain environment, although it sacrifices computational time to find more optimal solutions. Unlike CBAA or other conventional task allocation algorithms, the CAGA algorithm introduces adaptability and feasibility for real-world applications. In future works, plans are to use uncertainty synergistically while tackling the formation flight and task allocation problems and creating realistic environments much more analogous to real-world applications. Beyond the current uncertainty handling framework of the algorithm, incorporating additional checkpoints could enable more precise task assignments for agents. This enhancement can potentially improve efficiency, particularly in environments with high levels of uncertainty. Moreover, the failures that may arise due to the high communication burden of the system can be investigated, and the limited communication problem can be used jointly in the task assignment scenario. Also, an algorithm with higher efficiency and performance can be created by adapting the genetic parameters used in the proposed algorithm to the specific problem. Future research could explore trajectory planning problems for agents under uncertainty in conjunction with task assignments. Enhancing agents' adaptability and sensitivity in reaching tasks may lead to more efficient and robust decision-making in dynamic environments.

REFERENCES

Bischoff, E., Kohn, S., Hahn, D., Braun, C., Rothfuß, S., and Hohmann, S. (2024). Heuristic reoptimization of time-extended multi-robot task allocation problems. *Networks*, 84(1):64–83.

Bonandin, S. and Herty, M. (2024). Consensus-based algorithms for stochastic optimization problems. *arXiv preprint arXiv:2404.10372*.

Chen, Y., Zhang, Y., Zhang, G., and Gu, Y. (2024). A drone swarm-based wildfire search and rescue method with autonomous behavior modeling and centralized task assignment. In *2024 IEEE 4th International Conference on Power, Electronics and Computer Applications (ICPECA)*, pages 341–345. IEEE.

Choi, H.-L., Brunet, L., and How, J. P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4):912–926.

ElGibreen, H. and Youcef-Toumi, K. (2019). Dynamic task allocation in an uncertain environment with heterogeneous multi-agents. *Autonomous Robots*, 43:1639–1664.

Herty, M., Huang, Y., Kalise, D., and Kouhkouh, H. (2024). A multiscale consensus-based algorithm for multi-level optimization. *arXiv preprint arXiv:2407.09257*.

Hossain, M. S., Ibrahim, I., and Fukuta, N. (2023). Parallel task allocation in multi-robot environment under uncertainty based on auction mechanism. In *2023*

15th International Congress on Advanced Applied Informatics Winter (IIAI-AAI-Winter), pages 97–100. IEEE.

Hwang, N. E., Kim, H. J., and Kim, J. G. (2022). Centralized task allocation and alignment based on constraint table and alignment rules. *Applied Sciences*, 12(13):6780.

Li, Y., Zhang, Z., He, Z., and Sun, Q. (2024). A heuristic task allocation method based on overlapping coalition formation game for heterogeneous uavs. *IEEE Internet of Things Journal*.

Liu, B. and Liu, B. (2010). *Uncertainty theory*. Springer.

Liu, L. and Shell, D. A. (2013). Optimal market-based multi-robot task allocation via strategic pricing. In *Robotics: Science and Systems*, volume 9, pages 33–40.

Ozturk, H. B., Ozturk, U., Yavas, N. B., and Bingul, Z. (2024a). Development of optimal task allocation algorithm for unmanned aerial vehicle swarms. In *Proceedings of the Cognitive Models and Artificial Intelligence Conference*, pages 341–348.

Ozturk, H. B., Yavas, N. B., Ozturk, U., and Bingul, Z. (2024b). Game theory based decentralized multi-task allocation algorithm for multi-agent systems. In *2024 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–9. IEEE.

Peng, J., Viswanath, H., and Bera, A. (2024). Graph-based decentralized task allocation for multi-robot target localization. *IEEE Robotics and Automation Letters*.

Trigui, S., Koubaa, A., Cheikhrouhou, O., Youssef, H., Bennaceur, H., Sriti, M.-F., and Javed, Y. (2014). A distributed market-based algorithm for the multi-robot assignment problem. *Procedia Computer Science*, 32:1108–1114.

Wang, G., Wang, F., Wang, J., Li, M., Gai, L., and Xu, D. (2024). Collaborative target assignment problem for large-scale uav swarm based on two-stage greedy auction algorithm. *Aerospace Science and Technology*, 149:109146.

Yan, F. and Di, K. (2023). Solving the multi-robot task allocation with functional tasks based on a hyper-heuristic algorithm. *Applied Soft Computing*, 146:110628.

Zhang, Z., Jiang, J., Haiyan, X., and Zhang, W.-A. (2024). Distributed dynamic task allocation for unmanned aerial vehicle swarm systems: A networked evolutionary game-theoretic approach. *Chinese Journal of Aeronautics*, 37(6):182–204.