A Distributed Auction Algorithm for Task Assignment with Robot Coalitions

Ruiliang Deng, Rui Yan, *Member, IEEE,* Peinan Huang, Zongying Shi, *Member, IEEE,* and Yisheng Zhong

***Abstract*—This study addresses the task assignment problem with robot coalitions, as encountered in practical scenarios such as multiplayer reach-avoid games. Unlike the classical assignment problem where a single robot performs each task, the problem considered here involves tasks that require execution by a robot coalition consisting of two robots. This task assignment problem is a special instance of 3-set packing problem, which is known to be NP-hard. We introduce the concept of ϵ-coalition-competitive equilibrium (ϵ-CCE) to characterize a kind of approximate solution that offers guaranteed performance. A distributed auction algorithm is developed to find an ϵ-CCE within a finite number of iterations. Additionally, several enhancements have been implemented to adapt the auction algorithm for practical applications where the task assignment problem may vary over time. Numerical simulations demonstrate that the distributed algorithm achieves satisfactory approximation quality.**

***Index Terms*—Task assignment, robot coalitions, auction algorithm, competitive equilibrium, reach-avoid games.**

I. Introduction

TASK assignment plays a crucial role in numerous multi-robot applications, including transportation [1]–[4], surveillance [5], multi-robot routing [6]–[8], and multi-robot decision making [9]–[11]. In classical assignment problems, specifically, the problem of weighted matching in a bipartite graph [12], each task requires at most one robot to perform, and each robot can only perform at most one task. The objective is to find a one-to-one assignment of robots to tasks with the maximum payoff. The Hungarian algorithm has been widely used to efficiently solve such assignment problems in polynomial time [13]. Moreover, researchers have also proposed several distributed variants of the Hungarian algorithm to address scalability and the distributed nature of certain applications [14], [15].

However, in many practice scenarios, certain tasks may require multiple robots to perform cooperatively. One important field where such task assignment problems arise is in multiplayer pursuit-evasion games or reach-avoid games [16]–[19]. In these problems, a group of robots aims to capture as many evaders as possible before they enter specific regions. Each evader is treated as a task, and each robot is assigned to perform a particular task individually or in cooperation with other robots. The objective of the task assignment is to find an assignment that maximizes the number of captured evaders. According to the findings in [18]–[20], when individual robots are unable to perform a task, a *coalition* consisting of multiple robots can perform the task cooperatively. The number of robots required to successfully complete a task is typically bounded by the spatial dimensions of game scenarios [18]–[20]. For instance, in a two-dimensional environment, a task may require at most two robots. Consequently, the corresponding task assignment problem needs to consider assigning a coalition of multiple robots to a task. We refer to this problem as the *task assignment problem with robot coalitions*. In this paper, our focus is specifically on the scenario where a task requires a robot coalition consisting of a maximum of two robots.

The task assignment with robot coalitions can be viewed as a special instance of k-set packing problems [21], where k represents the maximum number of robots required to complete any task plus one. In a k-set packing problem, there is a given collection of sets, each with a size of at most k, and the objective is to find a maximal collection of disjoint sets. In the context of the task assignment problem, each set in the given collection corresponds to a pairing of a robot coalition and a task, with the coalition capable of completing the task. An assignment refers to a collection of disjoint sets. It is well known that k-set packing problems are NP-hard, and there has been a significant amount of research focused on developing approximation algorithms for these problems [21]–[24]. The most straightforward heuristic for the k-set packing problem is the greedy algorithm [22], shown to be a k-approximation algorithm, ensuring a solution with a payoff of at least $1/k$ of the optimal payoff. Other commonly used approximation algorithms are usually based on the local search method [22]–[24]. Specifically, a local search algorithm using swaps of size 2 can achieve an approximation ratio of $(k+1)/2$ [23]. By utilizing swaps of size r for larger values of r, the approximation ratio of a local search method can approach $k/2$. Further enhancements to the local search algorithm have the potential to yield an improved approximation ratio [23], [24]. However, it is important to note that these algorithms are centralized in nature. In many practical applications, running a centralized algorithm on a central server is not feasible due to limitations such as safety, robustness, and maintenance costs. In contrast, distributed algorithms offer scalability for both robots and tasks and exhibit superior capabilities in handling individual failures and dynamic environments. Consequently, they are better suited for autonomous task execution in complex environments. In order to cater to the needs of practical scenarios, this paper aims to address the development of a

This work was supported by the Science and Technology Innovation 2030-Key Project of "New Generation Artificial Intelligence" under Grant 2020AAA0108200.

R. Deng, P. Huang, Z. Shi and Y. Zhong are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: {drl20, hpn22}@mails.tsinghua.edu.cn; {szy, zys-dau}@mail.tsinghua.edu.cn).

R. Yan is with the School of Artificial Intelligence, Beihang University, Beijing, 100083, China (email: rui_yan@buaa.edu.cn).

distributed algorithm for the task assignment problem with robot coalitions.

Researchers have proposed various algorithms for distributed task assignment in different scenarios, broadly categorized as distributed Hungarian algorithm, market-based algorithms, game theoretical algorithms, and others. The distributed Hungarian algorithms [14], [15] are distributed variants of Hungarian algorithms, which are limited to classical task assignment problems where one task can be assigned to at most one robot. Market-based algorithms use market competition-like mechanisms for task allocation, for instance, the prevalent auction-based algorithms [25]–[33]. The auction algorithm in [25], [26] and its distributed variants [27] is designed for the classic task assignment problem and can provide an almost optimal solution. In [28], the authors extended the auction algorithm to solve the task assignment problem for grouped tasks. A group-based distributed auction algorithm was designed in [29] to solve a minimum time transportation problem which is NP-hard. While suitable for distributed implementation and providing performance guarantees for specific problems, these auction-based algorithms are tailored to particular problems and are not applicable to the task assignment problem with robot coalitions. Game theoretical algorithms [34], [35] convert task assignment problems into Nash equilibrium searching problems, but they might not converge within a finite time and are unsuitable for the task assignment problems in our purview. In addition, some consensus-based algorithms have also been utilized for task allocation problems distinct from the one under consideration, including the modified decentralized consensus-based bundle algorithm [36] and consensus alternating direction method of multipliers [37]. Moreover, researchers have utilized graph neural networks to fit the optimal assignment [38]. These algorithms possess advantages in terms of time complexity, but lack performance guarantees.

Since the k-set packing problem can be formulated as integer linear programs (ILPs), distributed algorithms for solving ILPs may be applicable to task assignment with robot coalitions. While an algorithm leveraging Lagrangian decomposition and distributed constraint satisfaction was presented in [39], it lacks convergence and optimality guarantees. Additionally, in [40], a quantized consensus algorithm was used for distributed task assignment, which cannot guarantee to converge in finite time. Recently, several distributed ILP algorithms have been developed that can guarantee convergence to optimal, near-optimal, or feasible solutions in finite time. Cutting-plane inspired algorithms were proposed in [41], and [42] introduced branch-and-price based approaches. Furthermore, dual decomposition, primal decomposition, and local computation based methods were proposed for mixed integer linear programs (MILPs) in [43]–[46]. However, it is important to note that while these algorithms ensure convergence in finite time, they are not polynomial-time algorithms.

To sum up, the current literature lacks polynomial-time distributed algorithms for the task assignment problem with robot coalitions. Addressing this gap, this paper introduces a distributed auction algorithm that can ensure an approximate solution with performance guarantees within a finite time. The primary contributions of this article can be summarized as follows:

1) We introduce the concept of ϵ-coalition-competitive equilibrium (ϵ-CCE), which is applicable in auction scenarios where multiple agents can form a coalition to compete for objects. We demonstrate that this equilibrium offers an approximate solution with guaranteed performance for the task assignment problem, allowing for the assignment of one task to a coalition comprising a maximum of two agents.
2) We propose a distributed auction algorithm that operates synchronously in rounds. This algorithm can find an ϵ-CCE within a finite number of rounds. We provide an upper bound on the total rounds before termination, which is linear in the number of robots and tasks for the unweighted versions of the task assignment problem.
3) We have introduced two enhancements to the distributed auction algorithm, facilitating its application in practical scenarios where robots need to execute actions concurrently with the algorithm execution, and in situations involving dynamic changes in task assignment problems.

The paper is structured as follows. Section II formulates the task assignment problem. Section III discusses several related integer linear programs. The coalition-competitive equilibrium is introduced in IV, and the distributed auction algorithm is proposed in Section V. Section VI presents improvements for practical implementation. Section VII evaluates the performance of the distributed auction algorithm. Finally, conclusions are included in Section VIII.

Notation: The set of positive integers is denoted as $\mathbb{N}^+$ Let $x \vee y = \max(x, y)$ for $x, y \in \mathbb{R}$. The n-dimensional all-ones vector is denoted as $\mathbf{1}_n$. For any real number x, the symbols $\lceil x \rceil$ and $\lfloor x \rfloor$ respectively denote the smallest integer greater than or equal to x and the largest integer smaller than or equal to x. The cardinality of a finite set $\mathcal{S}$ is represented by $|\mathcal{S}|$. The Cartesian product of two sets $\mathcal{A}$ and $\mathcal{B}$ is denoted by $\mathcal{A} \times \mathcal{B}$. In addition, for convenience of reference, we provide Tables I and II which display commonly used notations for both the problem statement and algorithm description.

II. Motivation and Problem Statement

A. Motivation

The task assignment with robot coalitions arises from multi-player reach-avoid games [16]–[20]. In a multi-player reach-avoid game, two teams of agents are involved: defenders and attackers. The attackers seek to enter a target region protected by the defenders, while the defenders aim to intercept as many attackers as possible before they reach the target. This game model has applications in various fields, such as attack and defense in robot soccer, maritime escort with unmanned surface vehicles, region defense and capture-the-flag in the military domain, and so on.

Due to the curse of dimensionality, solving multi-player reach-avoid games accurately is challenging. An effective strategy for the defense team is to divide defenders into disjoint subsets and assign attackers to these subsets [16]–[19].

This decomposes the original game into multiple easier-to-solve multi-vs-one reach-avoid games. To maximize successful defense, a special task assignment problem must be solved: each interception of an attacker is a task, requiring completion by a subset of defenders cooperatively. As shown in [18]–[20], for common two-dimensional multi-player reach-avoid games, each attacker needs at most two defenders for interception. Therefore, this paper focuses on task assignment problems in which each task requires at most two robots to complete cooperatively.

B. Problem Statement

Suppose that there are N_r robots and N_t tasks. The set of robots is represented as $\mathcal{R} = \{R_1, \ldots, R_{N_r}\}$, and the set of tasks is represented as $\mathcal{T} = \{T_1, \ldots, T_{N_t}\}$. When there is no ambiguity, the index will be used to represent the corresponding robot or task. In this paper, a subset of $\mathcal{R}$ with the size no more than two is called a **coalition**. The set of coalitions is represented as $\mathcal{A} = \{a \subset \mathcal{R} \mid |a| \leq 2\}$. An **assignment pair** is a pair of a coalition and a task. The assignment pair consisting of coalition a and task T_j is denoted as $(a, j) \in \mathcal{A} \times \mathcal{T}$. The assignment pair (a, j) is **feasible** if and only if the task T_j can be completed by the coalition a. The set of all feasible assignment pairs is denoted as $\mathcal{M} \subset \mathcal{A} \times \mathcal{T}$. Two assignment pairs with a common robot or task cannot be selected simultaneously. Thus, it is necessary to consider a conflict relation among feasible assignment pairs, which can be represented by a **conflict function** $c : \mathcal{M} \times \mathcal{M} \to \{0, 1\}$. For two pairs $m_1 = (a_1, j_1) \in \mathcal{M}$ and $m_2 = (a_2, j_2) \in \mathcal{M}$, $c(m_1, m_2) = 0$ if and only if these two assignment pairs do not conflict with each other, namely, $a_1 \cap a_2 = \emptyset$ and $j_1 \neq j_2$. Naturally, $c(m, m) = 1$ for any $m \in \mathcal{M}$. An assignment is a set of non-conflicting feasible assignments pairs defined as follows.

Definition 1 (Assignment)**.** *A set of feasible assignment pairs $\mathcal{S} \subset \mathcal{M}$ is an **assignment** if $c(m_1, m_2) = 0$ for all $m_1 \in \mathcal{S}$ and $m_2 \in \mathcal{S}$.*

Each feasible assignment pair is associated with a payoff. The payoffs are specified by a **payoff function** $v : \mathcal{M} \to \mathbb{R}$ such that $v(m) > 0$ for all $m \in \mathcal{M}$. The payoff of an assignment $\mathcal{S}$ is $v(\mathcal{S}) = \sum_{m \in \mathcal{S}} v(m)$, where the notation v is reused with no ambiguity. When $v(m) = 1$ for all $m \in \mathcal{M}$, the payoff of an assignment is its cardinality. The task assignment problem, consisting of the tuple $(\mathcal{A}, \mathcal{T}, \mathcal{M}, c, v)$, is formally stated as follows:

Task assignment with robot coalitions: Given the tuple $(\mathcal{A}, \mathcal{T}, \mathcal{M}, c, v)$, find an assignment $\mathcal{S}$ that achieves the highest payoff.

This task assignment problem is referred to as Γ. When $v(m) = 1, \ \forall m \in \mathcal{M}$, the task assignment problem aims to find the assignment that maximizes the number of tasks completed. This problem is referred to as Γ_0, and is called the *unweighted* task assignment with robot coalitions. In practical situations [17], [18], [47], [48], the focus is often solely on the number of tasks completed. Therefore, the goal of this paper is to solve the unweighted task assignment problem Γ_0.

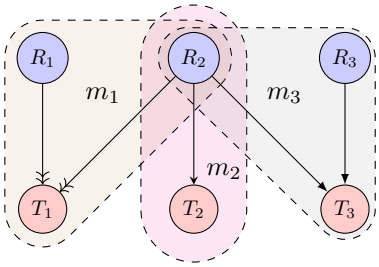


Fig. 1: An illustration of the task assignment problem Γ_0. The scenario involves three robots and three tasks. The coalition set $\mathcal{A}$ consists of the elements: $\{1, 2, 3, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$. The coalition $\{1, 2\}$ (resp., $\{2, 3\}$) can successfully perform task T_1 (resp., T_3), and the robot R_2 can complete task T_2 individually. Thus, there are three feasible pairs in $\mathcal{M}$: $m_1 = (12, 1)$, $m_2 = (2, 2)$, and $m_3 = (23, 3)$. It holds that $c(m_i, m_j) = 1$, when $i \neq j$, as all pairs involve R_2. Therefore, an optimal assignment would only include one of these three pairs.

The reason for introducing the problem Γ will be explained in Section III.

We next introduce several notations. First, a coalition consisting of a single robot R_i is denoted by its index i, and a coalition consisting of two distinct robots R_k and R_l, $k \neq l$, is denoted as $\{k, l\}$. Clearly, $\{k, l\}$ and $\{l, k\}$ represent the same coalition. We denote an assignment pair consisting of R_i and T_j as (i, j), and an assignment pair consisting of $\{k, l\}$ and T_j as (kl, j). The payoffs of (i, j) and (kl, j) are represented as $v(i, j)$ and $v(kl, j)$, respectively. We use the notation $R_i \in \mathcal{S}$ (resp., $T_j \in \mathcal{S}$) to indicate that R_i (resp., T_j) is present in an assignment pair in $\mathcal{S}$, and $R_i \notin \mathcal{S}$ (resp., $T_j \notin \mathcal{S}$) otherwise. Finally, the set of feasible pairs that do not contain robot or task involved in $\mathcal{S}$ is denoted as $\mathcal{M} \setminus \mathcal{S} \subset \mathcal{M}$. Several commonly used notations are summarised in Table I. An illustration of the task assignment problem Γ_0 is presented in Fig. 1.

TABLE I: Notations for task assignment problem

R_i	$\triangleq$	ith robot
$\mathcal{R}$	$\triangleq$	set of robots
T_j	$\triangleq$	jth task
$\mathcal{T}$	$\triangleq$	set of tasks
$\mathcal{A}$	$\triangleq$	set of coalitions
$\{k, l\}$	$\triangleq$	coalition involving R_k and R_l
$\mathcal{M}$	$\triangleq$	set of feasible assignment pairs
$c(\cdot, \cdot)$	$\triangleq$	conflict function
$v(\cdot)$	$\triangleq$	payoff function

Both the task assignment problems Γ_0 and Γ are **NP-hard** as shown in [18]. Hence, unless P=NP, we cannot obtain the optimal solution of the problem in polynomial time, and thus we need to consider polynomial-time approximation algorithms. Given a real number $k > 1$, an algorithm is called a ***k*-approximation** algorithm if it has approximation ratio k, namely, it can guarantee to find an assignment $\mathcal{S}$ such that

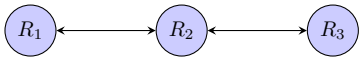


Fig. 2: A communication graph $\mathcal{G}$ for the example depicted in Fig. 1. In order to satisfy Assumption 1, the edge set $\mathcal{E}$ should include, at a minimum, the edges $(1,2)$ and $(2,3)$.

$v(\mathcal{S}^*) \leq kv(\mathcal{S})$, where $\mathcal{S}^*$ is an optimal assignment.

In this paper, we aim to address the task assignment problem Γ_0 in a distributed manner. For this purpose, we assume that robots can exchange information with each other in order to negotiate for an sub-optimal assignment. An undirected communication topology is considered in this paper. The communication network can be represented by an undirected graph $\mathcal{G} = (\mathcal{R}, \mathcal{E})$, where $\mathcal{E} \subset \mathcal{R} \times \mathcal{R}$ is the edge set. There is an edge (k,l) in $\mathcal{E}$ if and only if R_k and R_l can communicate with each other. The neighbor set of robot R_k is defined as $\mathcal{N}_k = \{R_l \in \mathcal{R} \mid (k,l) \in \mathcal{E}\}$. Fig. 2 provides an illustration of the communication graph. Regarding the communication topology, we make the following assumption.

Assumption 1. *If there exist coalitions $a_1, a_2 \in \mathcal{A}$ and task T_j such that $R_k \in a_1$, $R_l \in a_2$, $(a_1, j) \in \mathcal{M}$ and $(a_2, j) \in \mathcal{M}$, then $(k,l) \in \mathcal{E}$.*

Assumption 1 requires that robots capable of completing at least one common task, either individually or in cooperation with other robots, are able to communicate with each other. While this assumption may appear stringent, it is applicable in numerous practical scenarios. Particularly in certain open environments, the communication topology is primarily constrained by communication distance, and robots can only perform tasks within their perception range. If the communication distance exceeds twice the perception distance, robots capable of completing a specific task must be within each other's communication range. Consequently, Assumption 1 holds true in such cases. In [49], a similar assumption is made. If the assumption is not satisfied, the robots would need to engage in multi-hop communication to negotiate the allocation of a specific task, which would increase the number of messages exchanged.

We focus on synchronous distributed algorithms [50] in this paper. The development of asynchronous algorithms will be addressed in future work. Thus, we assume that all robots possess synchronous clocks, which can be realized through the use of clock synchronization methods. For example, satellite navigation systems can be used for clock synchronization in outdoor environments.

Our objective is to design a k-approximation distributed algorithm for the task assignment problem Γ_0 that runs in polynomial time. The approximation ratio k should be determined based on the trade-off between performance and complexity.

III. Related Integer Linear Programs

In this section, we present several ILPs that will be used in later sections.

The task assignment problem Γ can be encoded as an ILP. To achieve this, we introduce two indicator arrays $\mathbf{x} = (x_{i,j})_{(i,j)\in\mathcal{M}}$ and $\mathbf{y} = (y_{kl,j})_{(kl,j)\in\mathcal{M}}$ such that: $x_{i,j} = 1$ indicates that R_i is assigned to task T_j, and $x_{i,j} = 0$ otherwise; $y_{kl,j} = 1$ indicates that coalition $\{k,l\}$ is assigned to T_j, and $y_{kl,j} = 0$ otherwise. The ILP can be expressed as

$$\max_{\mathbf{x},\mathbf{y}} f(\mathbf{x},\mathbf{y}) = \sum_{(i,j)\in\mathcal{M}} v(i,j)x_{i,j} + \sum_{(kl,j)\in\mathcal{M}} v(kl,j)y_{kl,j} \tag{1a}$$

$$\text{s.t.} \quad \sum_{j} x_{i,j} + \sum_{i'\neq i,j} y_{ii',j} \leq 1, \quad \forall R_i \in \mathcal{R}; \tag{1b}$$

$$\sum_{i} x_{i,j} + \sum_{k<l} y_{kl,j} \leq 1, \quad \forall T_j \in \mathcal{T}; \tag{1c}$$

$$x_{i,j} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{M}; \tag{1d}$$

$$y_{kl,j} \in \{0,1\}, \quad \forall (kl,j) \in \mathcal{M}. \tag{1e}$$

The constraints (1b) and (1c) require that each robot or task appears in at most one assignment pair. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be an optimal solution to this ILP problem. An optimal assignment for Γ is given by $\mathcal{S}^* = \mathcal{S}_1^* \cup \mathcal{S}_2^*$, where

$$\mathcal{S}_1^* = \{(i,j) \in \mathcal{M} \mid x_{i,j}^* = 1\},$$
$$\mathcal{S}_2^* = \{(kl,j) \in \mathcal{M} \mid y_{kl,j}^* = 1\}.$$

For Γ_0, the objective function becomes

$$f_0(\mathbf{x},\mathbf{y}) = \sum_{(i,j)\in\mathcal{M}} x_{i,j} + \sum_{(kl,j)\in\mathcal{M}} y_{kl,j}.$$

As stated in Section II, the task assignment problem Γ_0 is our main focus. We can find an optimal or sub-optimal solution to Γ_0 by solving the problem Γ, which is supported by the following lemma.

Lemma 1. *Given sets $\mathcal{A}$, $\mathcal{T}$, $\mathcal{M}$ and function c. Assume that the function v satisfies that*

$$|v(m) - 1| < \frac{1}{2\min\{N_r, N_t\}}, \ \forall m \in \mathcal{M}. \tag{2}$$

If S^* is an optimal solution to Γ, then it is also an optimal solution to Γ_0.

Proof. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be an optimal solution to ILP (1) and $\mathcal{S}^*$ be the corresponding optimal assignment. If $\mathcal{S}^*$ is not an optimal solution to Γ_0, then there exists a feasible solution $(\mathbf{x}', \mathbf{y}')$ such that

$$f_0(\mathbf{x}', \mathbf{y}') \geq f_0(\mathbf{x}^*, \mathbf{y}^*) + 1.$$

According to (2), it holds that

$$\begin{aligned} &|f(\mathbf{x},\mathbf{y}) - f_0(\mathbf{x},\mathbf{y})| \\ &= \left| \sum_{(i,j)\in\mathcal{M}} (v(i,j)-1)x_{i,j} + \sum_{(kl,j)\in\mathcal{M}} (v(kl,j)-1)y_{kl,j} \right| \\ &< \frac{f_0(\mathbf{x},\mathbf{y})}{2\min\{N_r, N_t\}} \leq \frac{1}{2}. \end{aligned}$$

It follows that

$$f(\mathbf{x}', \mathbf{y}') > f_0(\mathbf{x}', \mathbf{y}') - \frac{1}{2} \geq f_0(\mathbf{x}^*, \mathbf{y}^*) + \frac{1}{2} > f(\mathbf{x}^*, \mathbf{y}^*),$$

which is a contradiction. Therefore, $\mathcal{S}^*$ is an optimal solution to Γ_0. □

According to Lemma 1, if we design the function v such that (2) holds, then an optimal solution to Γ also serves as an optimal solution to Γ_0. Consequently, we may obtain an approximate solution with guaranteed payoff to Γ_0 by solving the problem Γ approximately. Moreover, by setting v to generate marginally different payoffs within the set $\mathcal{M}$, we can reduce the competition for the same task among different coalitions, which can speed up the convergence of the distributed algorithm in the subsequent sections.

In a classical task assignment, each task can be performed by one robot. If we only consider the one-robot-one-task feasible pairs, the task assignment problem Γ becomes a classical one, which can be encoded as the following linear program (LP):

$$\begin{aligned} \max_{\mathbf{x}} \quad & g(\mathbf{x}) = \sum_{(i,j)\in\mathcal{M}} v(i,j)x_{i,j} \\ \text{s.t.} \quad & \sum_{j} x_{i,j} \leq 1, \ \forall R_i \in \mathcal{R}; \\ & \sum_{i} x_{i,j} \leq 1, \ \forall T_j \in \mathcal{T}; \\ & x_{i,j} \geq 0, \ \forall (i,j) \in \mathcal{M}. \end{aligned} \tag{3}$$

This LP can be derived from the ILP (1) via setting each element of $\mathbf{y}$ to zero and eliminating the integrality constraints on $\mathbf{x}$. It is well-known that the LP (3) has an optimal integer solution, which can be efficiently obtained in polynomial time [12]. Let g^* be the optimal value of the problem (3). Since every integer optimal solution to LP (3) is also feasible for the ILP (1), the optimal value of the ILP (1) is not less than g^*.

IV. Centralized Algorithms and Coalition-Competitive Equilibrium

The task assignment problem Γ_0 can be formulated as a 3-set packing problem [21], where one is given a collection of sets, each of size at most 3, and the aim is to find a collection of disjoint sets of maximum size. In Γ_0, the given collection of sets is $\mathcal{M}$, and a collection of disjoint sets is an assignment. The problem Γ can be regarded as a weighted version of the set packing problem. In this section, we will first briefly review several centralized algorithms for solving 3-set packing problems. Afterwards, we will generalize the concept of competitive equilibrium to the task assignment problem with robot coalitions which will be used in our distributed algorithm below.

A. Centralized Algorithms

The greedy algorithm is a natural heuristic for solving the set packing problem. When applied to the task assignment problem Γ, it works as shown in Algorithm 1. The algorithm proceeds by iteratively adding an assignment pair to the current solution (line 4). Subsequently, the added pair and all conflicting pairs are removed from consideration (line 5). According to [22], the greedy algorithm is a 3-approximation algorithm for Γ_0. If the selected pair m^* at line 3 possesses the maximum payoff, the greedy algorithm is also a 3-approximation algorithm for Γ [22]. Hence, the following result holds.

Algorithm 1: Centralized greedy algorithm

input : $\mathcal{M}$, v, c
output: An approximate solution $\mathcal{S}$ to Γ_0
1 $\tilde{\mathcal{M}} \leftarrow \mathcal{M}$, $\mathcal{S} \leftarrow \emptyset$;
2 **while** $\tilde{\mathcal{M}} \neq \emptyset$ **do**
3 Select an m^* in $\tilde{\mathcal{M}}$;
4 $\mathcal{S} \leftarrow \mathcal{S} \cup \{m^*\}$;
5 $\tilde{\mathcal{M}} \leftarrow \{m \in \tilde{\mathcal{M}} \mid c(m, m^*) = 0\}$.

Lemma 2. *Let $\mathcal{S}$ be an assignment and $\mathcal{S}^*$ be an optimal assignment for Γ_0. If for any given $m \in \mathcal{M}$ there exists an $m' \in \mathcal{S}$ such that $c(m, m') = 1$, then $|\mathcal{S}| \geq \frac{1}{3}|\mathcal{S}^*|$.*

Proof. The assignment $\mathcal{S}$ can be viewed as an output of the greedy algorithm 1 by successively selecting the pairs in $\mathcal{S}$ in any order. The conclusion appears to be evident. □

To achieve better approximation ratio, it is common to use local search methods [21], [23], [24]. Starting from an assignment, a local search method with size-r swaps iteratively replaces a subset of the current assignment of size smaller than r with a larger subset that does not conflict with the remainder of the assignment. The greedy algorithm can be seen as a simple local search method that performs size-one swaps. In particular, the local search method with size-two swaps is a 2-approximation algorithm for the problem Γ_0 [23].

B. Coalition-Competitive Equilibrium

The task assignment problem can be connected to the economic concept of competitive equilibrium (CE) [51]. The classical competitive equilibrium primarily focuses on assignment problems where each task is allocated to at most one robot. Before providing the distributed algorithm, we first introduce a generalized equilibrium concept which can be applied to the assignment problem Γ.

Imagine a market environment. Each task is associated with a price. The price vector is $\mathbf{p} = (p_1, \cdots, p_{N_t})$, where $p_j \geq 0$ is the price for task T_j. When tasks are allocated according to assignment $\mathcal{S}$, each robot can obtain a profit based on the price vector $\mathbf{p}$. The profit obtained from a task is equal to the payoff of the assignment pair minus the price of that task. If $(a, j) \in \mathcal{S}$, the robots in the coalition a will share the profit $v(a, j) - p_j$. In other words, $\sum_{i\in a} u_i = v(a, j) - p_j$, where u_i denotes the profit of R_i. A robot earns zero profit if it is not assigned to any task. The profit vector is denoted as $\mathbf{u} = (u_1, \cdots, u_{N_r})$.

First, we review the classical concept of competitive equilibrium. Assume that for all $(a, j) \in \mathcal{S}$, $|a| = 1$. In other words, each task is performed by only one robot. Then, the profit of a robot can be determined uniquely by the assignment $\mathcal{S}$ and the price vector $\mathbf{p}$. For a non-negative real number ϵ, the ϵ-competitive equilibrium (ϵ-CE) [51] is defined as follows:

Definition 2 (ϵ-CE)**.** *An assignment $\mathcal{S}$ and a price vector $\mathbf{p}$ are in ϵ-competitive equilibrium if:*

(a) $p_j = 0$, for all $T_j \notin \mathcal{S}$;
(b) for all $R_i \in \mathcal{R}$, it holds that

$$u_i + \epsilon \geq \max_{(i,j)\in\mathcal{M}} \left((v(i,j) - p_j) \vee 0\right),$$

where the profit vector $\mathbf{u}$ is determined by

$$u_i = \begin{cases} v(i,j) - p_j, & \text{if } \exists T_j, (i,j) \in \mathcal{S}; \\ 0, & \text{otherwise}. \end{cases}$$

The ϵ-CE is equivalent to the CE when $\epsilon = 0$. In CE, each robot is assigned to the task that maximizes its profit given the current prices. In ϵ-CE, no robot can increase its profit by more than ϵ by changing its current task.

In the problem Γ_0, there are tasks that are performed by coalitions of size two. The robots in a coalition can divide the total profit in different ways. Thus, the ϵ-CE is not applicable to the task assignment with robot coalitions. To address this, we define a generalized concept of competitive equilibrium that characterizes a steady status of task allocation where all robots are satisfied with the current assignment, considering the price vector.

Definition 3 (ϵ-CCE)**.** *An assignment $\mathcal{S}$, a price vector $\mathbf{p}$ and a profit vector $\mathbf{u}$ are in ϵ-coalition-competitive equilibrium (ϵ-CCE) if:*

(a) $p_j = 0,\ \forall T_j \notin \mathcal{S};\ u_i = 0,\ \forall R_i \notin \mathcal{S}$;
(b) for all $R_i \in \mathcal{R}$, it holds that

$$u_i + \epsilon \geq \max_{(i,j)\in\mathcal{M}} \left((v(i,j) - p_j) \vee 0\right);$$

(c) for every $(a,j) \in \mathcal{S}$, it holds that

$$\begin{aligned} \sum_{i\in a} u_i &= v(a,j) - p_j, \\ u_i \geq v(ik,j) - p_j,\ \forall i \in a,\ \forall R_k \notin \mathcal{S},\ &s.t.\ (ik,j) \in \mathcal{M}; \end{aligned} \tag{4}$$

(d) for $a \in \mathcal{A}$ such that $R_i \notin \mathcal{S}, \forall R_i \in a$, it holds that

$$0 \geq v(a,j) - p_j,\ \forall T_j \in \mathcal{T}, s.t.\ (a,j) \in \mathcal{M}.$$

The conditions (a) and (b) are the same as the conditions for defining ϵ-CE. According to (b), no robot has a sufficient incentive to change the current assignment status to perform any task individually, with the potential profit improvement limited to within ϵ. The equality in (c) states that the total profit within a coalition is equal to the payoff of the assignment pair minus the price of the task. The inequality in (c) indicates that once a robot is assigned, it lacks incentive to change the current coalition to form a new coalition with another robot which is not in $\mathcal{S}$. Furthermore, condition (d) states that unassigned robots cannot enhance their profits by perform any task in cooperation with other robots.

The ϵ-CCE offers solutions to Γ_0 with guaranteed performance. We assert that conditions (a) and (d) hold only if $\mathcal{M} \setminus \mathcal{S} = \emptyset$. Suppose both conditions are satisfied and there exists an assignment pair (a,j) in $\mathcal{M} \setminus \mathcal{S}$. It follows that $v(a,j) \leq p_j = 0$, which contradicts the assumption that $v(m) > 0$ for all $m \in \mathcal{M}$. Hence, the assertion is proven. Thus, the assumption of Lemma 2 is satisfied and the following lemma holds.

Lemma 3. *If an assignment $\mathcal{S}$ is in ϵ-CCE with a price vector $\mathbf{p}$ and a profit vector $\mathbf{u}$, then it holds that $|\mathcal{S}| \geq \frac{1}{3}|\mathcal{S}^*|$, where $\mathcal{S}^*$ is an optimal assignment for Γ_0.*

As shown in [51], the ϵ-CE offers approximate solutions to the classical task assignment problem, where each task is performed by one robot, with a deviation of no more than $N\epsilon$ from the optimal solution, where N is the number of robots. However, due to the NP-hardness of Γ_0, the ϵ-CCE cannot provide a similar performance approximation. Nevertheless, the ϵ-CCE still ensures a better performance compared to an approximate solution obtained by considering only the assignment pairs involving a single robot.

Lemma 4. *If an assignment $\mathcal{S}$, a price vector $\mathbf{p}$ and a profit vector $\mathbf{u}$ are in ϵ-CCE, then it holds that $v(\mathcal{S}) \geq g^* - N_s\epsilon$, where g^* is the optimal value of the problem* (3) *and $N_s = \min\{N_r, N_t\}$.*

Proof. The Lagrangian function of the LP (3) is

$$\begin{aligned} L(\mathbf{x},\mathbf{u},\mathbf{p}) = &\sum_{(i,j)\in\mathcal{M}} v(i,j)x_{i,j} + \sum_i u_i \left(1 - \sum_j x_{i,j}\right) \\ &+ \sum_j p_j \left(1 - \sum_i x_{i,j}\right) \\ = &\sum_i u_i + \sum_j p_j + \sum_{(i,j)\in\mathcal{M}} (v(i,j) - u_i - p_j)x_{i,j}. \end{aligned} \tag{5}$$

The dual problem can be expressed as

$$\begin{aligned} \min_{\mathbf{u},\mathbf{p}} \quad & \sum_i u_i + \sum_j p_j \\ \text{s.t.} \quad & u_i + p_j \geq v(i,j), \\ & u_i \geq 0,\ p_j \geq 0, \quad \forall R_i \in \mathcal{R},\ T_j \in \mathcal{T}. \end{aligned} \tag{6}$$

Let $\mathbf{x}$ be such that $x_{i,j} = 1$ if $(i,j) \in \mathcal{S}$ and $x_{i,j} = 0$ otherwise. Thus, $\mathbf{x}$ is a feasible solution to the LP (3). It holds that for $(i,j) \in \mathcal{S}$,

$$1 - \sum_{j'} x_{i,j'} = 1 - \sum_{i'} x_{i',j} = 0. \tag{7}$$

According to items (a) and (b) in Definition 3, if $\mathcal{S}$, $\mathbf{p}$ and $\mathbf{u}$ are in ϵ-CCE, then $(\mathbf{u} + \epsilon\mathbf{1}_{N_r}, \mathbf{p})$ and $(\mathbf{u}, \mathbf{p} + \epsilon\mathbf{1}_{N_t})$ are both feasible solutions to the dual problem (6). It follows that

$$\sum_i u_i + \sum_j p_j + N_s\epsilon \geq g^*. \tag{8}$$

According to item (c) in Definition 3, it is satisfied that

$$\sum_{(i,j)\in\mathcal{M}} (v(i,j) - u_i - p_j)x_{i,j} = 0. \tag{9}$$

According to (5), (7), (9) and item (c) in Definition 3, one has that

$$\begin{aligned}\sum_i u_i + \sum_j p_j &= \sum_{(i,j)\in\mathcal{M}} v(i,j)x_{i,j} + \sum_{(kl,j)\in\mathcal{S}} (u_k + u_l + p_j) \\ &= \sum_{(i,j)\in\mathcal{M}} v(i,j)x_{i,j} + \sum_{(kl,j)\in\mathcal{S}} v(kl,j) \\ &= \sum_{(a,j)\in\mathcal{S}} v(a,j) = v(\mathcal{S}).\end{aligned}$$

Therefore, it follows from (8) that $v(\mathcal{S}) \geq g^* - N_s\epsilon$. □

For Γ_0, both $v(\mathcal{S})$ and g^* are integers. If $N_s\epsilon < 1$, then it holds that $v(\mathcal{S}) \geq g^*$.

Corollary 1. *Assume that $v(m) = 1, \forall m \in \mathcal{M}$, and $N_s\epsilon < 1$. If an assignment $\mathcal{S}$, a price vector $\mathbf{p}$ and a profit vector $\mathbf{u}$ are in ϵ-CCE, then it holds that $v(\mathcal{S}) \geq g^*$.*

Remark 1. *Lemma 4 and Corollary 1 provide performance guarantees for worst-case scenarios. It is highly likely that the strict inequality holds in* (8)*. Thus, in ϵ-CCE, it is generally true that $v(\mathcal{S}) > g^*$ if ϵ is small. This implies that the ϵ-CCE offers superior solutions to Γ_0 compared to approximating the problem by only considering assignment pairs involving one robot. On the other hand, we cannot expect better results than Lemma 4 in the worst-case scenario, because the optimal value of Γ may be equal to g^*.*

V. Distributed Auction Algorithm

Employing market mechanisms to allocate tasks is an effective approach for achieving distributed task assignment. As discussed in Section IV, the market equilibrium ϵ-CCE can offer solutions to the task assignment problem Γ_0 with guaranteed performance. Thus, in this section, we propose a distributed auction algorithm that searches for an ϵ-CCE. This algorithm is adapted from Bertseka's auction algorithm for the classical assignment problem [26], [27], and it achieves a approximation ratio of 3, similar to the centralized greedy algorithm.

The algorithm runs synchronously in rounds, as depicted in Algorithm 2. Each round consists of three phases: a bidding phase (line 4), a market-clearing phase (line 5), and an informing phase (line 6). In the bidding phase, each unassigned robot makes a bid based on the price vector and profit information. The market-clearing phase determines the winner of the auctions. The winning robots subsequently share the new price information with their neighbors. Finally, during the informing phase, the unassigned robots calculate their profit estimates, which they subsequently send to their neighbors. If no more robots will bid after a certain round, then we say the algorithm has terminated.

The duration of each phase, denoted as τ, is a customizable parameter that needs to be set greater than the sum of the execution time and the maximum communication delay. For each robot, the operations involved in each phase primarily consist of arithmetic calculations and identifying maximum values from simple sets. We assume that an upper bound estimate for the communication delay can be obtained in practical application scenarios. Therefore, the parameter τ can be selected based on certain global information. Each round $n \in \mathbb{N}^+$ starts at t_n, and we assume that $t_n = t_0 + 3(n-1)\tau$.

Algorithm 2: Distributed Auction Algorithm for R_i

```
1 Init() ;                                   // at time t = t_0
2 n ← 1 ;
3 repeat:
4 |  Bid() ;                                 // at t = t_0 + 3(n-1)τ
5 |  ClearMarket() ;                         // at t = t_0 + (3n-2)τ
6 |  Inform() ;                              // at t = t_0 + (3n-1)τ
7 |  n ← n + 1 ;
```

Before diving into the details of the algorithm, we require several notations below. We use the variable s_i to represent the assignment status of R_i. A robot can be in one of three different assignment statuses: idle, bidding for a task, or being assigned. The corresponding labels for s_i are *'idle'*, *'bid'* and *'asg'*, respectively. The target task that R_i is assigned to or currently bidding for is denoted as e_i. If R_i has no target task, then $e_i = -1$. The robot with whom R_i intends to form or is forming a coalition is called the *partner* of R_i and is represented by c_i. If R_i has no partner, then $c_i = -1$. The set of feasible tasks of R_i is

$$\mathcal{T}_i = \{T_j \in \mathcal{T} \mid \exists a \in \mathcal{A},\ s.t.\ R_i \in a, (a,j) \in \mathcal{M}\},$$

which consists of tasks that R_i can complete individually or cooperatively. The set of tasks that R_i can perform individually is $\mathcal{T}_{i,\mathrm{s}} = \{T_j \in \mathcal{T}_i \mid (i,j) \in \mathcal{M}\}$. The set of unassigned neighbors of R_i at the beginning of round n, which are in the idle status, is denoted as $\bar{\mathcal{C}}_i^{(n)}$. Additionally, we define

$$\tilde{\mathcal{T}}_{i,\mathrm{c}} = \left\{(k,j) \mid R_k \in \bar{\mathcal{C}}_i^{(n)}, T_j \in \mathcal{T}_i, (ik,j) \in \mathcal{M}\right\}$$

to provide further clarity. Furthermore, the price vector at the beginning of round n is denoted as $\mathbf{p}^{(n-1)}$. The assignment at the beginning of round n is represented by $\mathcal{S}^{(n-1)}$, defined as:

$$\{(a,j) \mid \forall R_i \in a,\ s_i = \textit{'asg'},\ c_i \in a \cup \{-1\},\ e_i = j\}.$$

Moreover, we represent the subset of the assignment $\mathcal{S}^{(n-1)}$ that R_i has access to as

$$\mathcal{S}_i^{(n-1)} = \left\{(a,j) \in \mathcal{S}^{(n-1)} \mid \exists R_k \in \mathcal{N}_i \cup \{R_i\}, R_k \in a\right\}.$$

Finally, $(k\cdot, j) \in \mathcal{S}^{(n)}$ implies that there exists a robot R_l such that $(kl,j) \in \mathcal{S}^{(n)}$. Several frequently used notations are displayed in Table II.

Besides, communication is essential in the distributed auction algorithm. Two types of messages are used: bidding messages and informing messages. The bidding message and informing message send by R_i are denoted as M_{bid}^i and M_{inf}^i, respectively. The content in the message will be clarified in the context below.

TABLE II: Notations for distributed auction algorithm

s_i	$\triangleq$	assignment status of ith robot
e_i	$\triangleq$	target task of ith robot
c_i	$\triangleq$	partner of ith robot
$\mathcal{T}_i$	$\triangleq$	set of tasks that R_i can perform
$\bar{\mathcal{C}}_i^{(n)}$	$\triangleq$	set of unassigned neighbors of R_i at the beginning of round n
$p_j^{(n)}$	$\triangleq$	price of T_j at the end of round n
$u_i^{(n)}$	$\triangleq$	profit of R_i at the end of round n
$\mathcal{S}^{(n)}$	$\triangleq$	assignment at the end of round n
$\mathcal{S}_i^{(n)}$	$\triangleq$	subset of $\mathcal{S}_i^{(n)}$ available to R_i

A. Algorithm Description

Next, we will provide a detailed explanation of the distributed auction algorithm. We emphasize that we will describe the algorithm that *robot R_i needs to execute in round n*. Therefore, in all variables, the subscript i represents robot R_i, and the superscript n represents round n.

1) Initialization: When the algorithm begins at time t_0, the robot R_i needs to initialize the following variables:

$\mathcal{S}_i^{(0)}$: The assignment is empty as no robot is assigned;
$p_j^{(0)}$: It is initialised as $p_j^{(0)} = 0,\ \forall T_j \in \mathcal{T}_i$;
s_i: The assignment status of R_i is initialized as *'idle'*;
e_i and c_i: Initially, R_i does not have a target task or partner, so $e_i = -1$ and $c_i = -1$;
$u_i^{(0)}$: The initial profit is $u_i^{(0)} = \max_{(a,j)\in\mathcal{M}, i\in a} v(a,j)$;
$\bar{\mathcal{C}}_i^{(1)}$: It holds that $\bar{\mathcal{C}}_i^{(1)} = \mathcal{N}_i$.

The initialization subroutine is depicted in Algorithm 3.

Algorithm 3: $Init()$

$s_i \leftarrow$ *'idle'*, $e_i \leftarrow -1,\ c_i \leftarrow -1$;
$p_j^{(0)} \leftarrow 0,\ \forall T_j \in \mathcal{T}_i$;
$u_i^{(0)} \leftarrow \max_{(a,j)\in\mathcal{M}, i\in a} v(a,j)$;
$\mathcal{S}_i^{(0)} \leftarrow \emptyset,\ \bar{\mathcal{C}}_i^{(1)} = \mathcal{N}_i$;

The update of these variables will be described in detail in the variable update part at the end of this subsection.

2) The Bidding Phase: Algorithm 4 details the subroutine for the bidding phase. Table III provides a clear overview of the notations used in the algorithm. In this phase, if R_i is unassigned, it can select a feasible task that maximizes its profit and makes a bid accordingly. There are three feasible types of bids: solo bids, replacement bids and cooperative bids.

A solo bid is made for performing a task individually. The maximum profit that R_i can achieve by making a solo bid is

$$\hat{u}_{i,\mathrm{s}}^{(n)} = \max_{j\in\mathcal{T}_{i,\mathrm{s}}} \left(v(i,j) - p_j^{(n-1)} \right). \tag{10}$$

Algorithm 4: $Bid()$

```
/* Subroutines for the bidding phase */
Calculate according to (10)-(19).
if s_i = 'idle' then // bid only if being unassigned
    if max{û^(n)_{i,s}, û^(n)_{i,r}, û^(n)_{i,c}} > 0 then
        /* only bid for positive profit */
        if û^(n)_{i,s} ≥ û^(n)_{i,r} ∨ û^(n)_{i,c} then
            Calculate p̂_{e*_{i,s},i} according to (20);
            s_i ← 'bid', e_i ← e*_{i,s}, c_i ← −1,
              u_i^(n) ← v(i, e_i) − p̂_{e_i,i};
        else if û^(n)_{i,c} ≥ û^(n)_{i,r} then
            Calculate p̂_{e*_{i,c},i} according to (21);
            s_i ← 'bid', e_i ← e*_{i,c}, c_i ← c*_{i,c}, u_i^(n) ←
              0.5 (û_i^(n) + v(ic_i, e_i) − p̂_{e_i,i} − û_{c_i}^(n));
        else
            Calculate p̂_{e*_{i,r},i} according to (22);
            s_i ← 'bid', e_i ← e*_{i,r}, c_i ← c*_{i,r},
              u_i^(n) ← v(ic_i, e_i) − p̂_{e_i,i} − u_{c_i}^(n−1);
        M^i_bid ← (i, e_i, c_i, p̂_{e_i,i});
        Send the message M^i_bid to neighbors.
    else
        u_i^(n) ← u_i^(n−1).
```

TABLE III: Notations for bidding phase

$\hat{u}_{i,\mathrm{s}}^{(n)}$	$\triangleq$	highest profit of solo bids
$e_{i,\mathrm{s}}^*$	$\triangleq$	task that achieves profit $\hat{u}_{i,\mathrm{s}}^{(n)}$
$\check{u}_{i,\mathrm{s}}^{(n)}$	$\triangleq$	second-highest profit of solo bids
$\hat{u}_{i,\mathrm{r}}^{(n)}$	$\triangleq$	highest profit of replacement bids
$c_{i,\mathrm{r}}^*$	$\triangleq$	partner that achieves profit $\hat{u}_{i,\mathrm{r}}^{(n)}$
$e_{i,\mathrm{r}}^*$	$\triangleq$	task that achieves profit $\hat{u}_{i,\mathrm{r}}^{(n)}$
$\check{u}_{i,\mathrm{r}}^{(n)}$	$\triangleq$	second-highest profit of replacement bids
$\hat{u}_i^{(n)}$	$\triangleq$	highest profit of solo and replacement bids
$\hat{u}_{i,\mathrm{c}}^{(n)}$	$\triangleq$	highest profit of cooperative bids
$c_{i,\mathrm{c}}^*$	$\triangleq$	partner that achieves profit $\hat{u}_{i,\mathrm{c}}^{(n)}$
$e_{i,\mathrm{c}}^*$	$\triangleq$	task that achieves profit $\hat{u}_{i,\mathrm{c}}^{(n)}$
$\hat{p}_{j,i}$	$\triangleq$	bidding price for task T_j

The optimal task $T_{e_{i,\mathrm{s}}^*}$ can be chosen according to

$$e_{i,\mathrm{s}}^* \in \arg\max_{j\in\mathcal{T}_{i,\mathrm{s}}} \left(v(i,j) - p_j^{(n-1)} \right). \tag{11}$$

The second-highest profit of solo bids is

$$\check{u}_{i,\mathrm{s}}^{(n)} = \max_{j\in\mathcal{T}_{i,\mathrm{s}}\setminus\{e_{i,\mathrm{s}}^*\}} \left(v(i,j) - p_j^{(n-1)} \right). \tag{12}$$

It is worth noting that $\check{u}_{i,\mathrm{s}}^{(n)}$ may be equal to $\hat{u}_{i,\mathrm{s}}^{(n)}$.

A replacement bid is made to request a substitute for the partner of an assigned neighbor. For example, R_i can bid to perform $T_j \in \mathcal{T}_i$ cooperating with $R_k \in \mathcal{N}_i$ if there exists $a \in \mathcal{A}$ such that $k \in a$ and $(a, j) \in \mathcal{S}_i^{(n-1)}$. The incorporation of replacement bids aims to leverage the potential for enhancing the payoff of the assignment by altering partners. We assume that the profit of a robot remains unchanged after its partner is replaced. The optimal robot $R_{k^*_{i,\mathrm{r}}}$ and task $T_{e^*_{i,\mathrm{r}}}$ for a replacement bid satisfy that

$$(c^*_{i,\mathrm{r}}, e^*_{i,\mathrm{r}}) \in \underset{(k\cdot,j)\in\mathcal{S}_i^{(n-1)}}{\arg\max} \left(v(ik, j) - u_k^{(n-1)} - p_j^{(n-1)} \right), \tag{13}$$

and the maximum profit is given by

$$\hat{u}_{i,\mathrm{r}}^{(n)} = v(ic^*_{i,\mathrm{r}}, e^*_{i,\mathrm{r}}) - u_{c^*_{i,\mathrm{r}}}^{(n-1)} - p_{e^*_{i,\mathrm{r}}}^{(n-1)}. \tag{14}$$

The second-highest profit for such a bid is

$$\check{u}_{i,\mathrm{r}}^{(n)} = \max_{(k\cdot,j)\in\mathcal{S}_{\mathrm{r}}} \left(v(ik, j) - u_k^{(n-1)} - p_j^{(n-1)} \right), \tag{15}$$

where $\mathcal{S}_{\mathrm{r}} = \mathcal{S}_i^{(n-1)} \setminus \{(a, e^*_{i,\mathrm{r}}) \mid c^*_{i,\mathrm{r}} \in a\}$. The maximum profit of solo and replacement bids is defined as

$$\hat{u}_i^{(n)} = \hat{u}_{i,\mathrm{s}}^{(n)} \vee \hat{u}_{i,\mathrm{r}}^{(n)}. \tag{16}$$

A cooperative bid is made when the robot attempts to perform a task cooperatively with an unassigned neighbor. The optimal pair of partner $R_{c^*_{i,\mathrm{c}}}$ and task $T_{e^*_{i,\mathrm{c}}}$ satisfies

$$(c^*_{i,\mathrm{c}}, e^*_{i,\mathrm{c}}) \in \tilde{\mathcal{T}}_i^* = \underset{(k,j)\in\tilde{\mathcal{T}}_{i,\mathrm{c}}}{\arg\max} \left(v(ik, j) - \hat{u}_k^{(n)} - p_j^{(n-1)} \right), \tag{17}$$

and the optimal profit for a cooperative bid is

$$\hat{u}_{i,\mathrm{c}}^{(n)} = v(ic^*_{i,\mathrm{c}}, e^*_{i,\mathrm{c}}) - \hat{u}_{c^*_{i,\mathrm{c}}}^{(n)} - p_{e^*_{i,\mathrm{c}}}^{(n-1)}. \tag{18}$$

The variable $\hat{u}_k^{(n)}$ serves as an estimate for the profit to be allocated to R_k. It should be noted that when $n \geq 2$, $\bar{\mathcal{C}}_i^{(n)}$ and $\hat{u}_k^{(n)},\ k \in \bar{\mathcal{C}}_i^{(n)}$ can be derived from the informing messages in the last round, which will be discussed in detail after the explanation for the informing phase. However, when $n = 1$, R_i does not have access to the value of $\hat{u}_k^{(1)}$, and therefore an estimated value $\max_{j\in\mathcal{T}_i,(ik,j)\in\mathcal{M}} v(ik, j)$ is used. When the optimal pair is not unique, the optimal partner and target are chosen as follows:

$$c^*_{i,\mathrm{c}} = \max_{(k,j)\in\tilde{\mathcal{T}}_i^*} k, \quad e^*_{i,\mathrm{c}} = \max_{(c^*_{i,\mathrm{c}},j)\in\tilde{\mathcal{T}}_i^*} j. \tag{19}$$

The robot R_i has an incentive to make a bid only when there exists a bid type that yields a positive profit. In other words, at least one of $\hat{u}_{i,\mathrm{s}}^{(n)}$, $\hat{u}_{i,\mathrm{r}}^{(n)}$ and $\hat{u}_{i,\mathrm{c}}^{(n)}$ is positive (line 3 in Algorithm 4). The robot will select the bid type with the highest profit according to a specific priority order (lines 4-12). Solo bids have the highest priority (lines 4-6), followed by cooperative bids (lines 7-9). When R_i decides to make a bid, its assignment status will be set to *'bid'*, and its target task e_i and partner c_i will be determined according to (11), (13) or (17), depending on the selected bid type. In particular, c_i is set as -1 if a solo bid is made.

The robot needs to provide a bidding price when making a bid. In order to maximize the chance of winning the auction, the bidding price is set to be the highest possible value, ensuring that the robot cannot increase its profit by more than ϵ through changing the task or partner, or changing the bidding type to solo or replacement bids. If the solo bid is selected, the bidding price is determined by

$$\hat{p}_{e^*_{i,\mathrm{s}},i} = v(i, e^*_{i,\mathrm{s}}) - \left(\left(\check{u}_{i,\mathrm{s}}^{(n)} \vee \hat{u}_{i,\mathrm{r}}^{(n)} \right) \vee 0 \right) + \epsilon. \tag{20}$$

The price ensures that the robot obtains a profit that is not lower than the second highest profit of a solo bid or the highest profit of a replacement bid, by more than ϵ. For a cooperative bid, the bidding price is determined by

$$\hat{p}_{e^*_{i,\mathrm{c}},i} = v(ic^*_{i,\mathrm{c}}, e^*_{i,\mathrm{c}}) - \left(\left(\hat{u}_i^{(n)} + \hat{u}_{c^*_{i,\mathrm{c}}}^{(n)} \right) \vee \check{u}_{i,\mathrm{c}} \right) + \epsilon, \tag{21}$$

where

$$\check{u}_{i,\mathrm{c}} = \max_{j\neq e^*_{i,\mathrm{c}},\, (ic^*_{i,\mathrm{c}},j)\in\mathcal{M}} \left(v(ic^*_{i,\mathrm{c}}, j) - p_j^{(n-1)} \right) \vee 0.$$

This is the highest price, ensuring that R_i has no incentive to perform another task with the same partner or submit a bid of a different type, plus ϵ. In the case of a replacement bid, the bidding price is determined as follows:

$$\hat{p}_{e^*_{i,\mathrm{r}},i} = v(ic^*_{i,\mathrm{r}}, e^*_{i,\mathrm{r}}) - u_{c^*_{i,\mathrm{r}}}^{(n-1)} - \left(\left(\hat{u}_{i,\mathrm{s}}^{(n)} \vee \check{u}_{i,\mathrm{r}}^{(n)} \right) \vee 0 \right) + \epsilon. \tag{22}$$

After calculating the bidding price, R_i updates its profit accordingly to satisfy the equality in (4), as shown in lines 6, 9 and 12 of Algorithm 4.

Remark 2. *If only solo bids are taken into account, the distributed auction algorithm depicted in Algorithm 2 will be degenerated to Bertseka's auction algorithm for the classical assignment problem [26], [27]. The inclusion of replacement bids and cooperative bids is crucial for utilizing assignment pairs that involve two robots, to obtain a solution that surpasses considering only pairs involving a single robot. These bids incorporate the idea of local search into Bertseka's auction algorithm. Through a cooperative bid (or a solo bid), a robot coalition comprising two robots (or a single robot) not yet assigned can be assigned to perform an unassigned task. This resembles an iteration of the greedy algorithm. Additionally, bids for assigned tasks potentially lead to assigned robots becoming unassigned during a round. If these robots make bids and become assigned again in subsequent rounds, the overall process can be viewed as a local search swap of size 2.*

The bidding information is shared with neighbors through bidding messages (line 14). The bidding message sent by R_i is of the form $M_{bid}^i = (i, e_i, c_i, \hat{p}_{e_i,i})$.

3) The Market-Clearing Phase: The subroutine for the market-clearing phase is described in Algorithm 5. In this phase, if R_i has a target task, namely, $e_i \geq 0$, it participates in the auction to assign a coalition to perform e_i. Let $\mathcal{R}_{bid}^{(n)}$ represent the set of robots who have made a bid in the bidding phase of round n. The set of bids in the auction for e_i is $\mathcal{B}(e_i) = \mathcal{B}_{\mathrm{s}}(e_i) \cup \mathcal{B}_{\mathrm{r}}(e_i) \cup \mathcal{B}_{\mathrm{c}}(e_i)$, where

$$\mathcal{B}_{\mathrm{s}}(e_i) = \{(k, \hat{p}_{e_i,k}) \mid k \in \mathcal{R}_{bid}^{(n)},\ e_k = e_i,\ c_k = -1\},$$

$$\mathcal{B}_{\mathrm{r}}(e_i) = \{(kc_k, \hat{p}_{e_i,k}) \mid k \in \mathcal{R}_{bid}^{(n)},\ e_k = e_i,\ c_k \in \mathcal{S}^{(n-1)}\},$$

and

$$\mathcal{B}_{\mathrm{c}}(e_i) = \{(kl, \hat{p}_{e_i,k}) \mid k < l,\ k,l \in \mathcal{R}_{bid}^{(n)},\ e_k = e_l = e_i,\ c_k = l,\ c_l = k\}$$

representing the set of solo, replacement, and cooperative bids, respectively. A cooperative bid is considered in the auction only if both robots in the coalition choose the same assignment pair for bidding, as indicated in the definition of $\mathcal{B}_{\mathrm{c}}(e_i)$. According to Assumption 1, R_i can receive all necessary bidding messages to determine $\mathcal{B}(e_i)$.

Algorithm 5: $ClearMarket()$

1 **if** $s_i =$ *'bid'* **then** // unassigned in last round
2 $(a^*, p^*) \leftarrow Winner(\mathcal{B}(e_i))$.
3 **if** $R_i \in a^*$ **then** // win the auction for e_i
4 $s_i \leftarrow$ *'asg'*, $p_{e_i}^{(n)} \leftarrow p^*$;
5 $M_{inf}^i \leftarrow \left(i, s_i, e_i, c_i, p_{e_i}^{(n)}, u_i^{(n)}\right)$;
6 Send the message M_{inf}^i to neighbors.
7 **else** // lose the auction for e_i
8 $s_i \leftarrow$ *'idle'*, $e_i \leftarrow -1$, $c_i \leftarrow -1$, $u_i^{(n)} \leftarrow 0$.
9 **else if** $s_i =$ *'asg'* **then** // assigned in last round
10 **if** $\mathcal{B}(e_i) \neq \emptyset$ **then**
11 $(a^*, p^*) \leftarrow Winner(\mathcal{B}(e_i))$, $p_{e_i}^{(n)} \leftarrow p^*$.
12 **if** $R_i \in a^*$ **then** // win the auction for e_i
13 $c_i \leftarrow a^* \setminus R_i$, $u_i^{(n)} \leftarrow u_i^{(n-1)}$;
14 $M_{inf}^i \leftarrow \left(i, s_i, e_i, c_i, p_{e_i}^{(n)}, u_i^{(n)}\right)$;
15 **else** // lose the auction for e_i
16 $s_i \leftarrow$ *'idle'*, $e_i \leftarrow -1$, $c_i \leftarrow -1$, $u_i^{(n)} \leftarrow 0$;
17 $M_{inf}^i \leftarrow (i, s_i, -1, -1, 0, 0)$;
18 Send the message M_{inf}^i to neighbors.
19 **else**
20 $u_i^{(n)} \leftarrow u_i^{(n-1)}$.

Robot R_i utilizes Algorithm 6 to determine the winning bid of the auction (lines 2 and 11 of Algorithm 5). The winner must have the highest bidding price for e_i (line 2 of Algorithm 6). When there is more than one bid that has the highest bidding price, a priority order is used to resolve the tie (line 3). To establish this priority order, a function $Pri : \mathcal{A} \rightarrow \mathbb{R}$ is employed, where a higher value indicates a higher priority. In this study, it is assumed that a coalition consisting of one robot holds a higher priority compared to a coalition consisting of two robots. For a coalition $a \in \mathcal{A}$, let $I_R(a)$ denote the maximum index among the robots included, and let $I_L(a)$ denote the minimum index. The priorities of two coalitions with the same number of robots can be compared using the following rules: if $I_R(a_1) > I_R(a_2)$, then $Pri(a_1) > Pri(a_2)$; if $I_R(a_1) = I_R(a_2)$, then $Pri(a_1) > Pri(a_2)$ when $I_L(a_1) > I_L(a_2)$. It should be noted that a priority order can be established based on different rules.

The winning bid of the auction is represented by (a^*, p^*). If R_i wins the auction, or in other words, if $\mathcal{B}(e_i) \neq \emptyset$ and $R_i \in a^*$, its assignment status will be updated to *'asg'* (line 4 of Algorithm 5). Otherwise, the assignment status s_i will be set as *'idle'* (lines 8 and 16). The price of e_i is updated to the bidding price of the winner, p^*. In case R_i was previously assigned in the last round, its partner will be changed after winning the auction, while its profit remains unchanged (line 13).

Algorithm 6: $(a^*, p^*) = Winner(\mathcal{B}(e_i))$

1 **if** $\mathcal{B}(e_i) \neq \emptyset$ **then**
2 $\mathcal{B}^* \leftarrow \arg\max_{(a,p)\in\mathcal{B}(e_i)} p$.
3 $(a^*, p^*) \leftarrow \arg\max_{(a,p)\in\mathcal{B}^*} Pri(a)$.
4 **else**
5 $(a^*, p^*) \leftarrow (\emptyset, 0)$.

If R_i wins the auction, it should share the new price and profit with its neighbors by sending an informing message (lines 5 and 14). In the event that R_i loses the auction but was assigned in the last round, it is also required to inform its neighbors about its new status (line 17). The informing message sent by R_i is of the form $M_{inf}^i = \left(i, s_i, e_i, c_i, p_{e_i}^{(n)}, u_i^{(n)}\right)$.

4) The Informing Phase: The subroutine for the informing phase is presented in Algorithm 7. According to the informing messages, robot R_i can update the price $p_j^{(n)}$ for task $T_j \in \mathcal{T}_i$ and the profit $u_k^{(n)}$ for its assigned neighbor R_k. If R_i is unassigned, it needs to calculate a profit estimate $\hat{u}_i^{(n+1)}$ according to (10) and (14). The obtained profit estimate will then be sent to neighbors through an informing message (line 4 in Algorithm 7). This profit estimate will be used by its unassigned neighbors when making a cooperative bid.

Algorithm 7: $Inform()$

1 Update prices and profits according to received informing messages.
2 **if** $s_i =$ *'idle'* **then**
3 Calculate $\hat{u}_i^{(n+1)}$ according to (10), (14) and (16);
4 $M_{inf}^i \leftarrow \left(i, s_i, -1, -1, 0, \hat{u}_i^{(n+1)}\right)$;
5 Send the message M_{inf}^i to neighbors.

5) Variable Update: Robot R_i updates its stored variables based on the informing messages it receives. Its neighbors' assignment statuses, target tasks and partners can be directly obtained from the informing messages. Information about neighbors who do not send informing messages in this round remains unchanged. For any task $T_j \in \mathcal{T}_i$, the coalition assigned to perform it and the price $p_j^{(n)}$ can be derived from the informing messages sent during the market-clearing phase. If there are no informing messages about T_j, then the assigned coalition and price for T_j remain unchanged in this round. As a result, after round n, R_i can determine the assignment pairs in $\mathcal{S}_i^{(n)}$. The set $\bar{\mathcal{C}}_i^{(n+1)}$ consists of R_i's neighbors who have sent informing messages in the informing phase of round n. The value $\hat{u}_k^{(n+1)}$ for $k \in \bar{\mathcal{C}}_i^{(n+1)}$ can be obtained from these informing messages.

B. Analysis of the Distributed Auction Algorithm

In this subsection, the performance of the distribute auction algorithm is analyzed. It is shown that the algorithm terminates at an ϵ-CCE in a finite number of rounds, as stated in Theorems 1 and 2. It is worth noting that termination of the algorithm occurs when no further bids are made by robots after a certain round.

Theorem 1. *The distributed auction algorithm terminates in a finite number of rounds. It requires a maximum of* $N_s \left\lceil \frac{\max_{m\in\mathcal{M}} v(m)}{\epsilon} \right\rceil$ *rounds before the termination, where* $N_s = \min\{N_r, N_t\}$.

Proof. If no bidding messages are generated at round n, it implies that $\max\left\{\hat{u}_{i,\mathrm{s}}^{(n)}, \hat{u}_{i,\mathrm{r}}^{(n)}, \hat{u}_{i,\mathrm{c}}^{(n)}\right\} = 0$ for each $R_i \notin \mathcal{S}^{(n-1)}$. After round n, there will be no changes in prices and profits, indicating that the algorithm terminates. It can be seen that if $T_j \in \mathcal{S}^{(n)}$, then $T_j \in \mathcal{S}^{(n+k)}$ for any positive integer k. Thus, if the prices of N_s tasks are not lower than $\max_{m\in\mathcal{M}} v(m)$, then every robot is either assigned or would not benefit from making a bid, resulting in the termination of the algorithm.

Assume that algorithm terminates at round n_e. At round $n < n_e$, there must exist $T_j \in \mathcal{T}$ such that $\mathcal{B}(j) \neq \emptyset$. To see this, notice that there is at least one robot who makes a bid. If there is a solo bid or a replacement bid, it is obvious that $\mathcal{B}(j) \neq \emptyset$ for certain T_j. Otherwise, there are only cooperative bids. Consider the set of robots defined as follows:

$$\mathcal{R}_c^* = \underset{R_i\in\mathcal{R}_{bid}^{(n)}}{\arg\max} \left(v(ic_i, e_i) - \hat{u}_i^{(n)} - \hat{u}_{c_i}^{(n)} - p_{e_i}^{(n-1)} \right).$$

Let R_{i^*} be such that $i^* = \max_{i\in\mathcal{R}_c^*} i$. It can be proved that $c_{c_{i^*}} = i^*$ and $e_{c_{i^*}} = e_{i^*}$, implying that $\mathcal{B}(e_{i^*}) \neq \emptyset$. Since $R_{i^*} \in \mathcal{R}_c^*$, it holds that for any $R_k \notin \mathcal{S}^{(n-1)}$ and any $T_j \in \mathcal{T}_{c_{i^*}}$,

$$\begin{aligned} v(i^* c_{i^*}, e_{i^*}) - \hat{u}_{i^*}^{(n)} - \hat{u}_{c_{i^*}}^{(n)} - p_{e_{i^*}}^{(n-1)} \\ \geq v(c_{i^*}k, j) - \hat{u}_{c_{i^*}}^{(n)} - \hat{u}_k^{(n)} - p_j^{(n-1)}. \end{aligned}$$

This implies that

$$v(i^* c_{i^*}, e_{i^*}) - \hat{u}_{i^*}^{(n)} - p_{e_{i^*}}^{(n-1)} \geq v(c_{i^*}k, j) - \hat{u}_k^{(n)} - p_j^{(n-1)}.$$

Thus, $(i^*, e_{i^*}) \in \tilde{\mathcal{T}}_{c_{i^*}}^*$, where $\tilde{\mathcal{T}}_i^*$ is defined by (17). Let $k^* = \max_{(k,j)\in\tilde{\mathcal{T}}_{c_{i^*}}^*} k$. It must hold that $i^* = k^*$. Otherwise, $k^* > i^*$ and $k^* \in \mathcal{R}_c^*$ which contradicts the definition of i^*. Additionally, according to (19), we have

$$e_{i^*} = \min_{(c_{i^*},j)\in\tilde{\mathcal{T}}_i^*} j = \min_{(i^*,j)\in\tilde{\mathcal{T}}_{c_{i^*}}^*} j.$$

Hence, we conclude that $c_{c_{i^*}} = i^*$ and $e_{c_{i^*}} = e_{i^*}$.

According to line 4, the price of at least one task will be updated at round $n < n_e$. It can be easily proven, based on equations (20), (21) and (22), that an updated price increases by at least ϵ. On the other hand, the price of an task will not increase if it is already not lower than $\max_{m\in\mathcal{M}} v(m)$. Therefore, after a maximum of $N_s \left\lceil \frac{\max_{m\in\mathcal{M}} v(m)}{\epsilon} \right\rceil$ rounds, there will be at least N_s tasks with a price that is not lower than $\max_{m\in\mathcal{M}} v(m)$. This indicates that the algorithm has terminated. □

The distributed auction algorithm terminates after a maximum of $3\tau N_s \left\lceil \frac{\max_{m\in\mathcal{M}} v(m)}{\epsilon} \right\rceil$ time. The factors that affect this upper bound on the termination time include not only the number of robots and tasks, but also the payoff function v, the duration τ, and the parameter ϵ.

We can demonstrate that Algorithm 2 operates as a polynomial time algorithm when addressing the considered problem Γ_0. In Γ_0, the payoff function is unity. The parameter ϵ can be set to a fixed value. As a result, the factor multiplied by N_s in Theorem 1 becomes $\lceil 1/\epsilon \rceil$, which is independent of both the input's size and value. This implies that the algorithm terminates after a maximum number of rounds linear with N_s. Next, we analyze the factor τ. As previously mentioned, the duration τ depends on both the communication delay and the computational time at each phase. The computational time can be directly analyzed as follows. During the bidding phase, each unassigned robot needs to perform calculations according to equations (10)-(22). These calculations involve conducting algebraic operations on each element within sets that contain a maximum of $N_r N_t$ elements, as well as identifying the maximum elements in these sets. During the market-clearing phase, robots need to determine the winner of each auction by utilizing Algorithm 6, which involves finding the maximum element of a set of size no more than N_r. The computational complexity during the informing phase is the same as that during the bidding phase. Thus, the computational complexity for each robot at each phase is at most $O(N_r N_t)$. Consequently, the computational time complexity of the distributed auction algorithm is $O(N_s N_r N_t)$. On the other hand, analyzing the maximum communication delay exceeds the scope of this paper. Yet, it is reasonable to assume a polynomial bound of order c for the maximum communication delay concerning the number of robots. Therefore, the duration τ is polynomial in the number of robots and tasks. The total time complexity of the distributed auction algorithm is $O(N_s(N_r N_t + N_r^c))$. Therefore, the auction algorithm is a polynomial time algorithm for the unweighted task assignment problem with robot coalitions.

For the general weighted task assignment problem Γ, the algorithm proposed is merely a pseudo-polynomial time algorithm, as its time complexity depends on the maximum payoff among all feasible pairs. It is worthy to note one exceptional case, which is elaborated on in Section III: an approximate solution to an unweighted task assignment problem can be attained by addressing a weighted one. Under this circumstance, an assumption must be made that the inequality (2) is valid. Consequently, the payoff function is bounded by 2, independent of the input. In the context of such special weighted task assignment problems, the auction algorithm can also be regarded as polynomial time.

Theorem 2. *The distributed auction algorithm terminates at an ϵ-CCE.*

Proof. Assume that the algorithm terminates at round n_e. Thus, no bidding message will be generated after the n_eth round. The final assignment is denoted as $\mathcal{S} = \mathcal{S}^{(n_e)}$. We also denote $\mathbf{u} = \mathbf{u}^{(n_e)}$ and $\mathbf{p} = \mathbf{p}^{(n_e)}$. The items outlined in

Definition 3 will be checked one by one.

(a) It is obvious that $u_i = 0$ if $s_i =$ *'idle'*, namely, $R_i \notin \mathcal{S}$. Once a task is assigned in a particular round, it must continue to be assigned in the subsequent rounds. It follows that if $p_j > 0$, then $T_j \in \mathcal{S}$. Hence, we can conclude that $p_j = 0$ for $T_j \notin \mathcal{S}$.

(b) If $R_i \notin \mathcal{S}$, then we have:

$$u_i = 0 \geq \hat{u}_i^{(n_e)} \geq \max_{(i,j)\in\mathcal{M}} v(i,j) - p_j.$$

On the other hand, if R_i is assigned, let n_i be the index of the last round at which R_i makes a bid. In this case, we have $u_i = u_i^{(n_i)}$ and $p_{e_i} \geq p_{e_i}^{(n_i)}$. According to the update rules for $u_i^{(n)}$ (in lines 6, 9, 12 and 13 of Algorithm 4), and the expressions of $\hat{p}_{e_i,i}$, we can prove that $u_i^{(n_i)} + \epsilon \geq \check{u}_{i,\mathrm{s}}^{(n_i)} \vee 0$ if $c_i = -1$, and $u_i^{(n_i)} + \epsilon \geq \hat{u}_{i,\mathrm{s}}^{(n_i)} \vee 0$ otherwise. In the former case, we have

$$u_i^{(n_i)} = v(i,e_i) - p_{e_i}^{(n_i)} \geq \max_{j\in\mathcal{T}_{i,\mathrm{s}}\setminus\{e_i\}} v(i,j) - p_j^{(n_i-1)} - \epsilon.$$

In the latter case, we have

$$u_i^{(n_i)} + \epsilon \geq \max_{(i,j)\in\mathcal{M}} v(i,j) - p_j^{(n_i-1)}.$$

It follows that item (b) is satisfied because the price does not decrease.

(c) The equation can be derived directly from the update rule specified in lines 6, 9 and 12 of Algorithm 4. If the inequality does not hold, then there exists a robot $R_k \notin \mathcal{S}$ for which $\hat{u}_{k,\mathrm{r}}^{(n_e+1)} \geq v(ki,j) - u_i^{(n_e)} - p_j^{(n_e)} > 0$. This contradicts the assumption that no bidding message will be generated after round n_e. Thus, the inequality holds true when the algorithm terminates.

(d) The inequality can be proven using a contradiction argument, similar to the proof for item (c).

In conclusion, the distributed auction algorithm terminates at an ϵ-CCE. □

The parameter ϵ is crucial for the performance of the distributed auction algorithm. Theorem 1 implies that the maximum running time is inversely proportional to ϵ. A larger value of ϵ results in faster termination. When $\epsilon \geq \max_{m\in\mathcal{M}} v(m)$, the algorithm terminates within a maximum of N_s rounds. Conversely, a smaller value of ϵ is more likely to yield a better solution, as indicated by the findings of Lemma 4 and Corollary 1. Since solutions to Γ_0 have integer payoffs, it is sufficient to ensure that $N_s\epsilon < 1$. If we set the parameter ϵ to vary with the size of problem such that $\epsilon \approx \frac{1}{N_s}$, the running time is at most $3\tau N_s^2 \max_{m\in\mathcal{M}} \lceil v(m) \rceil$. Additionally, the algorithm's time complexity derived previously will increase by a factor of N_s. In practice, if the assignment pairs involving a single robot dominate in $\mathcal{M}$, then it is advisable to choose a small value of ϵ to uphold the guarantee stated in Corollary 1. However, if there are constraints on the running time, opting for a large value of ϵ would lead to faster termination.

According to Lemma 3 and Theorem 2, the distributed auction algorithm is a 3-approximation algorithm for the problem Γ_0. Although this result may seem unsatisfactory, the simulation results presented in Section VII demonstrate that despite its sub-optimal worst-case guarantee, the algorithm performs very well on average. Additionally, the simulation results indicate that the average performance of the distributed auction algorithm proposed in this paper is not significantly different from a 2-approximation centralized local search algorithm.

Remark 3. *According to existing studies on approximation algorithms for set packing problems [21], [23], [24], designing distributed algorithms with a better approximation ratio may rely on local search methods with swaps of size 2 or more. Implementing such a local search method requires unanimous agreement among multiple robots for a feasible swap related to more than one task. To this end, robots need to gather non-local information (e.g., assignment pairs that do not involve them) and take non-local decisions (e.g., deciding on the allocation of tasks that do not relate to them). Such an algorithm would require more communication and would likely involve centralized decision-making. In contrast, the proposed distributed algorithm only requires each robot to be aware of the assignment pairs involving it and to make decisions only for its own target task. As the proposed algorithm already exhibits satisfactory average performance in simulation, we do not attempt to directly design a distributed version of the local search method.*

Remark 4. *In cases where Assumption 1 does not hold, we can extend the distributed auction algorithm by modifying each phase to consist of Δ communication rounds, where Δ represents the maximum network diameter. Upon receiving the messages, the robots immediately forward them as necessary. Consequently, at the beginning of the next phase, each robot is able to acquire the messages required to execute the auction algorithm. The performance of the modified algorithm remains unchanged, while the maximum number of communication rounds required for termination becomes $3N_s\Delta \left\lceil \frac{\max_{m\in\mathcal{M}} v(m)}{\epsilon} \right\rceil$.*

VI. Improvements for Practice Implementation

When applying the distributed auction algorithm to practical scenarios, there are several challenges that must be considered. Firstly, robots may need to execute assigned tasks before the task assignment algorithm completes. Consequently, the tasks allocated to individual robots are subject to frequent changes, leading to frequent switches in their controls. Secondly, the task assignment problem may not be static, as the tuple $(\mathcal{A}, \mathcal{T}, \mathcal{M}, c, v)$ can change over time. To tackle these challenges, we provide several improvements for the distributed auction algorithm.

A. Handling Frequent Task Switches

In practice, robot R_i can maintain two additional variables, $\bar{c}_i$ and $\bar{e}_i$, which represent its actual partner and task, respectively. The robot makes real-time decisions based on $\bar{c}_i$ (if $\bar{c}_i \geq 0$) and $\bar{e}_i$. On the other hand, the variables c_i and e_i are solely used in the execution of the distributed auction algorithm. The robot R_i has the flexibility to choose any pair $(a,j) \in \mathcal{M}$ where $i \in a$ in order to initialize $\bar{c}_i$ and $\bar{e}_i$. The

values of $\bar{c}_i$ and $\bar{e}_i$ are updated to c_i and e_i, respectively, at a lower frequency compared to $1/3\tau$. If the robot is not assigned at the update moment, then $\bar{c}_i$ and $\bar{e}_i$ will remain unchanged. This approach helps mitigate or even eliminate frequent switches in controls, without modifying the auction algorithm.

B. Handling Time-Varying Assignment

To apply the auction algorithm for a time-varying scenario, two improvements are required to guarantee that the algorithm always finds a feasible solution. These improvements will be explained from the perspective of robot R_i:

1) *Information Update:* R_i updates the information about the tuple $(\mathcal{A}, \mathcal{E}, \mathcal{M}, c, v)$ only at the beginning of the market-clearing phase. This can avoid inconsistent information used by different robots during the bidding phase (note that when making a bid, a robot uses the information sent by neighbors during the market-clearing phase and informing phase of the last round). Specifically, robot R_i needs to update the sets $\mathcal{T}_i$ and $\mathcal{N}_i$, the subset of $\mathcal{M}$ associated with R_i, and the information about the function v.
2) *Release Infeasible Task:* It is necessary to release the assignment pairs that become infeasible due to changes in the set $\mathcal{M}$. After determining the auction winner in the market-clearing phase, if R_i is assigned, it needs to check the feasibility of the current assignment pair. Let m_i be (i, e_i) if $c_i < 0$ and (ic_i, e_i) otherwise. If m_i still exists in the set $\mathcal{M}$, then the robot has no further action except to send an informing message. However, if the current assignment pair m_i is no longer feasible due to changes in the problem, robot R_i should release the task e_i. The assignment status s_i will be set to *'idle'*, and both e_i and c_i will be set to -1. A release message, taking the form of $M^i_{rls} = (i, e_i, c_i)$, will be sent to neighbors to notify them of the release. Upon receiving the release message, the neighbors will be aware that R_i and its partner are now unassigned, as is the task e_i. *The price of e_i will be reset to zero*. If R_i is unassigned after determining the auction winner, it has nothing to do in this phase.

Next, we discuss the performance of the distributed auction algorithm for the time-varying task assignment problem. Since the problem is time-varying, we can only consider the termination of the algorithm within a specific time period. Let $\mathcal{K}$ represent the set of times when the set $\mathcal{M}$ changes. In practice, it is reasonable to assume that $\mathcal{K}$ is a finite set. The elements of $\mathcal{K}$ are $\kappa_1 < \kappa_2 < \cdots < \kappa_{|\mathcal{K}|}$. During the time interval $[\kappa_k, \kappa_{k+1})$, the set $\mathcal{M}$ remains unchanged. According to Theorem 1, if we design the payoff function such that (2) always holds, then the termination of the distributed auction algorithm can be guaranteed in the time interval $[\kappa_k, \kappa_{k+1})$ if $|\kappa_{k+1} - \kappa_k| \geq 3\tau N_s \left\lceil \frac{2}{\epsilon} \right\rceil$. In fact, set $\mathcal{M}$ rarely undergoes significant changes, and the auction algorithm can proceed using the existing feasible solution. Therefore, termination is often achieved within few rounds. In a time-varying scenario, the distributed auction algorithm does not guarantee termination at an ϵ-CCE. This is because conditions (b) and (c) in Definition 3 become difficult to satisfy due to changes in the payoff function and the release of infeasible assignment pairs. However, it is still ensured that each unassigned robot (or task) has a zero profit (or price), satisfying condition (a) in Definition 3. Additionally, condition (d) is met; otherwise, the algorithm would not have terminated. Therefore, according to the proof for Lemma 3, the auction algorithm can still guarantee an approximation ratio of 3.

VII. Simulation

In this section, we evaluate the effectiveness of the distributed auction algorithm through numerical simulations. We begin by testing the algorithm's performance on fixed task assignment problems. Subsequently, we apply the auction algorithm to a reach-avoid game problem.

A. Time-Invariant Task Assignment

We first conducted a series of tests to evaluate the performance of the auction algorithm on the time-invariant task assignment problem Γ_0. In order to comprehensively evaluate the performance of the algorithm, the impact of three factors are considered: the number of robots and tasks, indicated by N_s; the average number of assignment pairs per task, indicated by $\rho = \frac{|\mathcal{M}|}{N_s}$; and the proportion of assignment pairs involving two robots among $\mathcal{M}$, indicated by η (if there are M_c assignment pairs involving two robots, then $\eta = \frac{M_c}{|\mathcal{M}|}$). Additionally, we evaluated the algorithm's performance for different value of the parameter ϵ. In each situation, we carried out ten thousand random tests, with the tuple $(\mathcal{A}, \mathcal{T}, \mathcal{M}, c, v)$ randomly generated while ensuring the specified values for N_s, ρ, and η. In these tests, we take $N_t = N_r = N_s$. The payoff function v was designed to satisfy the inequality (2). We calculated the mean and standard deviation of the number of phases before the algorithm terminated, as well as the mean and standard deviation of the payoff ratio between the optimal solution and the solution found by the algorithm (namely, $|\mathcal{S}^*|/|\mathcal{S}|$), based on the ten thousand tests. The optimal payoff employed in the payoff ratio calculation is obtained by solving the ILP (1) using the mixed-integer programming solver in the cvxpy library.

In our evaluation, we compared the performance of our distributed auction algorithm with that of two centralized algorithms. The first centralized algorithm is a 3-approximation algorithm, which employs a greedy algorithm with an initial feasible solution obtained by solving the ILP (3). The second centralized algorithm utilizes a local search method with size-two swaps, with an optimal solution to ILP (3) as the initial solution, which is a 2-approximation algorithm. We calculated the average payoff ratio of the solutions found by these centralized algorithms.

Below, we will discuss the impact of different factors on algorithm performance separately. When testing each factor, the other factors take values in the middle of the ranges we are focusing on.

TABLE IV: Payoff ratios with different N_s.

N_s	AUC		GRE		LOC	
	mean	std	mean	std	mean	std
10	1.0065	0.0285	1.0237	0.0509	1.0100	0.0345
25	1.0132	0.0229	1.0327	0.0332	1.0181	0.0253
50	1.0154	0.0167	1.0359	0.0235	1.0210	0.0180
75	1.0164	0.0139	1.0370	0.0194	1.0223	0.0148
100	1.0169	0.0122	1.0378	0.0168	1.0229	0.0129

TABLE V: Number of phases with different N_s.

	$N_s = 10$	$N_s = 25$	$N_s = 50$	$N_s = 75$	$N_s = 100$
mean	10.433	13.767	17.097	19.251	20.758
std	10.238	9.129	9.348	10.129	8.399

TABLE VI: Payoff ratios with different ρ.

ρ	AUC		GRE		LOC	
	mean	std	mean	std	mean	std
2	1.0233	0.0240	1.0509	0.0334	1.0165	0.0196
4	1.0149	0.0166	1.0359	0.0236	1.0209	0.0180
6	1.0034	0.0083	1.0098	0.0136	1.0078	0.0118
8	1.0003	0.0025	1.0013	0.0053	1.0011	0.0048
10	1.0000	0.0006	1.0001	0.0016	1.0001	0.0014

TABLE VII: Number of phases with different ρ.

	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$	$\rho = 10$
mean	8.505	16.620	44.099	38.058	39.529
std	2.435	9.054	140.705	40.153	20.176

1) Impact of N_s: We set the value of ρ to 4 and the value of η to 0.5. The parameter ϵ was set as 0.02. We conducted tests for different values of N_s, specifically 10, 25, 50, 75, and 100.

Table IV displays the payoff ratios for three algorithms: the distributed auction algorithm (abbreviated as AUC), the 3-approximation centralized algorithm (abbreviated as GRE), and the 2-approximation centralized algorithm (abbreviated as LOC). The standard deviations of the ratios are given in the 'std' columns. It is observed that the distributed auction algorithm demonstrates varying payoff ratios ranging from 1.0065 to 1.0169 as N_s increases from 10 to 100. Similar increases in payoff ratios are observed for the two centralized algorithms. Notably, the auction algorithm exhibits a significantly superior average payoff ratio compared to the 3-approximation centralized algorithm, while its average payoff is comparable to the 2-approximation centralized algorithm. Surprisingly, for the specific value of ρ and η used in the tests, the auction algorithm even outperforms the 2-approximation centralized algorithm slightly.

The number of phases the auction algorithm runs before termination is shown in Table V. As the value of N_s increases, the number of phases also increases, which aligns with the conclusion of Theorem 1.

2) Impact of ρ: To investigate the impact of the average number of assignment pairs par task, we set N_s to 50 and η to 0.5, while the parameter ϵ was fixed at 0.02. The tests were performed with different values of ρ, namely, 2, 4, 6, 8, and 10. The results are summarized in Table VI and Table VII.

The increased value of ρ corresponds to a higher occurrence of redundant assignment pairs, thereby increasing the probability of finding a favorable solution using an approximate algorithm. It is evident from the results that the payoff ratios typically decrease with increasing values of ρ. Remarkably, our auction algorithm demonstrates superior performance compared to both centralized algorithms when dealing with larger values of ρ.

As observed in Table VII, the number of phases required for the algorithm to terminate does not exhibit a monotonic relationship with ρ. This behavior can be attributed to two factors. On one hand, the increasing value of ρ means that more assignment pairs need to be searched, which leads to an increase in time complexity. On the other hand, increasing ρ reduces the competition between different coalitions, allowing for faster termination.

3) Impact of η: We conducted tests with fixed values of $N_s = 50$ and $\rho = 4$. The algorithm was tested using $\epsilon = 0.02$, with η varying from 0 to 1. A value of $\eta = 0$ means that all pairs involves only one robot, while $\eta = 1$ indicates that all pairs involve two robots. The payoff ratios and number of phases are shown in Table VIII and Table IX, respectively.

For problems where all pairs involve only one robot, optimal solutions can be obtained in polynomial time. As the value of η increases, finding a solution that closely approximates the optimal solution becomes more challenging, resulting in larger payoff ratios. The auction algorithm outperforms both centralized algorithms in the case of small values of η. Even for large values of η, the auction algorithm maintains a superior average payoff compared to the 3-approximation centralized algorithm.

When there are more pairs involving two robots, the auction algorithm terminates in fewer phases. This can be partly attributed to a reduction in the number of assignment pairs included in the optimal assignment, as η increases. However, the relationship between the number of running phases and η is not monotonic, as indicated in Table IX.

4) Impact of parameter ϵ: We investigated the impact of ϵ with fixed values of $N_s = 50$, $\rho = 4$ and $\eta = 0.5$. Four values of ϵ were tested: 0.02, 0.1, 0.5, and 1.1. The obtained

TABLE VIII: Payoff ratios with different η.

η	AUC		GRE		LOC	
	mean	std	mean	std	mean	std
0	1	0	1	0	1	0
0.25	1.0020	0.0064	1.0048	0.0099	1.0040	0.0090
0.5	1.0149	0.0166	1.0359	0.0236	1.0209	0.0180
0.75	1.0470	0.0272	1.0811	0.0338	1.0337	0.0224
1	1.1274	0.0484	1.1676	0.0574	1.0599	0.0218

TABLE IX: Number of phases with different η.

	$\eta = 0$	$\eta = 0.25$	$\eta = 0.5$	$\eta = 0.75$	$\eta = 1$
mean	38.901	50.843	16.620	11.105	14.496
std	29.678	161.775	9.054	3.055	3.721

TABLE X: Payoff ratios with different ϵ.

	$\epsilon = 0.02$	$\epsilon = 0.1$	$\epsilon = 0.5$	$\epsilon = 1.1$
mean	1.0149	1.0153	1.0220	1.0880
std	0.0166	0.0167	0.0192	0.0364

results are displayed in Table X and Table XI. Increasing the value of ϵ leads to higher payoff ratios and a decrease in the number of required phases. These observations align with the discussion presented in Subsection V-B.

In addition to the aforementioned tests, we compared the computational efficiency of the distributed auction algorithm with two centralized algorithms. The results of this comparison are depicted in Figure 3, showing the average computational time based on 100 random experiments. We conducted this test using Python 3.11 on a PC equipped with a 12th Gen Intel Core i5 processor running at 3.70GHz, coupled with 16GB of RAM. We set $\rho = 8$ and $\eta = 0.5$ for this test, and the parameter ϵ was set to 0.02. It is important to note that communication delays were not included in this analysis, and the reported computation time for the auction algorithm represents the total computation time across all robots. As evident from the figure, the total computation time of the auction algorithm falls between that of the greedy algorithm and the local search method. Notably, the average computation time per robot, which corresponds to $1/N_r$ of the total computation time, is significantly lower than that of the centralized algorithms.

We can now provide a summary of the performance of the proposed distributed auction algorithm. Despite being a 3-approximation algorithm, the auction algorithm demonstrates an average payoff ratio close to 1 for problem instances with $N_s \leq 100$. The auction algorithm outperforms the 3-approximation centralized algorithm in all the conducted tests. In certain cases, the auction algorithm even surpasses the 2-approximation algorithm, particularly when there is a high average number of assignment pairs per task or a low proportion of assignment pairs involving two robots. In almost all scenarios, the auction algorithm exhibits a payoff ratio comparable to the 2-approximation centralized algorithm, except when $\eta = 1$. Roughly speaking, larger values of N_s and ρ or smaller values of η, require more phases for the auction

TABLE XI: Number of phases with different ϵ.

	$\epsilon = 0.02$	$\epsilon = 0.1$	$\epsilon = 0.5$	$\epsilon = 1.1$
mean	16.620	16.941	14.190	7.701
std	9.054	7.384	4.167	1.554

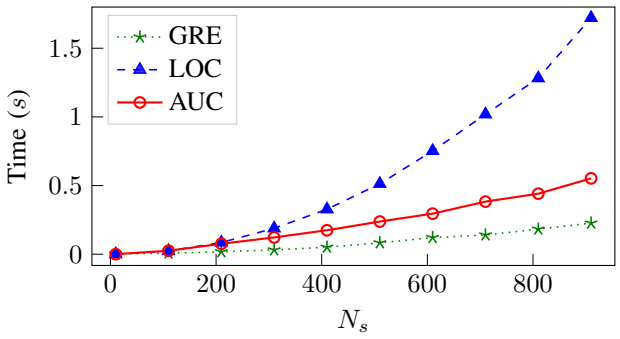


Fig. 3: Comparison of computational running times between the distributed auction algorithm and two centralized algorithms. The factors ρ and η are set to 8 and 0.5, respectively. The parameter ϵ is set to 0.02.

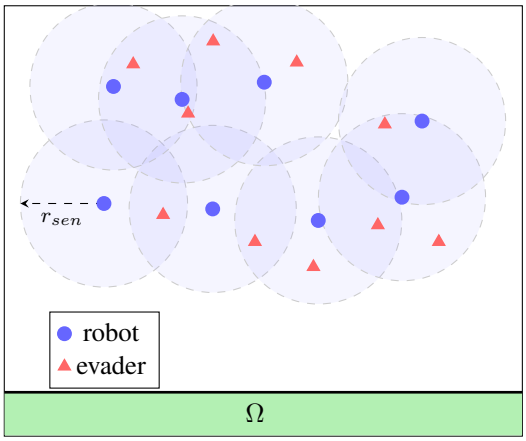
algorithm to terminates. On average, the number of required phases typically ranges from 10 to 50 (approximately 3-20 rounds). However, for specific values of ρ and η, the algorithm may require hundreds number of phases to terminate. In cases where running time becomes a constraint, it is necessary to choose a larger value of ϵ to expedite termination.

B. Reach-Avoid Games

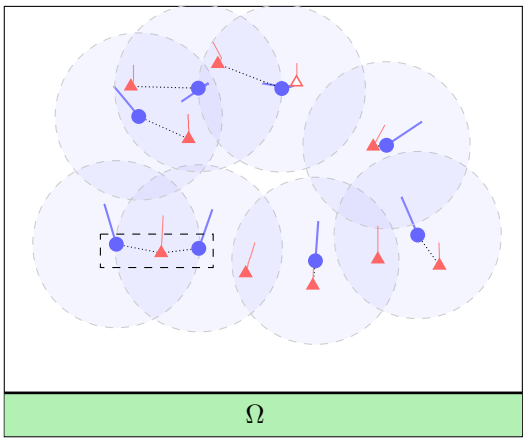
We proceed to utilize the distributed auction algorithm in a time-varying task assignment problem that arises from a multiplayer reach-avoid game scenario [10], [11], [17]–[19], incorporating the improvements discussed in Section VI. In the reach-avoid game, we have a total of $N_r = 8$ robots tasked with protecting a half plane *target region* Ω in a two-dimensional space from the intrusion of $N_t = 10$ evaders. Each evader corresponds to a task. The robots have a sensing range of $r_{sen} = 100\,m$, through which they can detect the presence of evaders. Furthermore, robots can communicate with each other if their distance does not exceed $r_{com} = 220\,m$. By setting the parameters such that $r_{com} > 2r_{sen}$, we ensure that Assumption 1 is satisfied. Both the robots and evaders move with simple motion dynamics, allowing them to instantaneously adjust their velocities within a specified maximum speed. The robots' maximum speeds range from $9\,m/s$ to $11\,m/s$, while the evaders' maximum speeds range from $4\,m/s$ to $9\,m/s$. When the distance between an evader and a robot is less than $5\,m$, the evader is captured and subsequently removed from the game. On the other hand, once an evader enters the target region Ω, robots are no longer allowed to pursue it. An illustration of the game problem is depicted in Fig. 4(a).

The robots aim to find an assignment that maximizes the number of evaders they can capture. According to the results in [18]–[20], a maximum coalition of two robots is required to capture each evader in this game problem. Thus, the task assignment problem can be modeled as Γ_0. The task set $\mathcal{T}$ corresponds to the set of evaders. An assignment pair $(a, j) \in \mathcal{A} \times \mathcal{T}$ is feasible if:

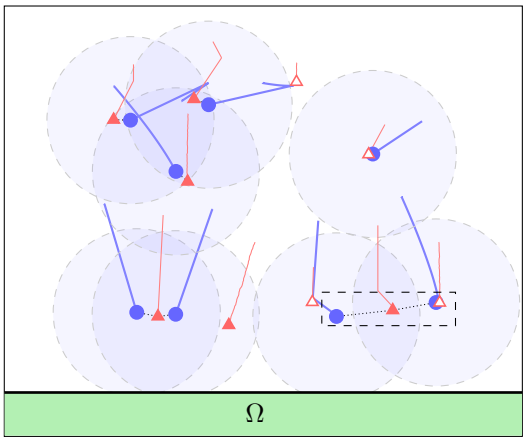
1) The coalition a can guarantee to capture the evader j before it enters Ω. This can be determined based on the solution to the game of kind [18], [19];



(a) $t = 0\,s$



(b) $t = 5\,s$



(c) $t = 13.5\,s$

Fig. 4: Snapshots of the game process. The robots are represented by blue circles while the evaders are depicted as red triangles. The observation range of the robots are illustrated using light blue circles enclosed by dashed lines. The trajectories of the robots and evaders are depicted by blue and red curves, respectively. The assignment obtained through the auction algorithm is indicated by dotted lines connecting the relevant robots and evaders. Additionally, the removed evaders are shown with hollow triangles.

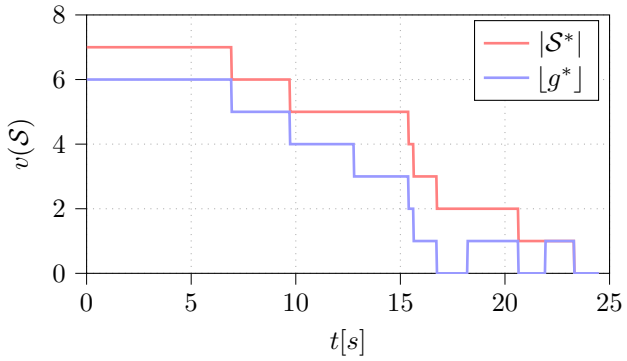
2) The robots in coalition a have the ability to observe the evader j;
3) If there are two robots in a, they can communicate with each other.

The set $\mathcal{M}$ consists of all assignment pairs that satisfy the above conditions at the current time. We design the function v according to the solution to the game of degree [18], ensuring that $v(m) - 1 > 0,\ \forall m \in \mathcal{M}$ and inequality (2) holds. We employ the distributed auction algorithm to search for a solution to Γ_0. The robots adopt strategies according to the obtained assignment. The time duration τ is set to $0.01\,s$, which is reasonable in many practical scenarios. Additionally, we set the parameter ϵ to 0.1. Considering that the sets $\mathcal{T}$ and $\mathcal{M}$ change whenever an evader is captured or enters the region Ω, we incorporate the improvements discussed in Section VI. To avoid frequent task switches, the robots update their actual tasks and partners at intervals of $0.1\,s$.

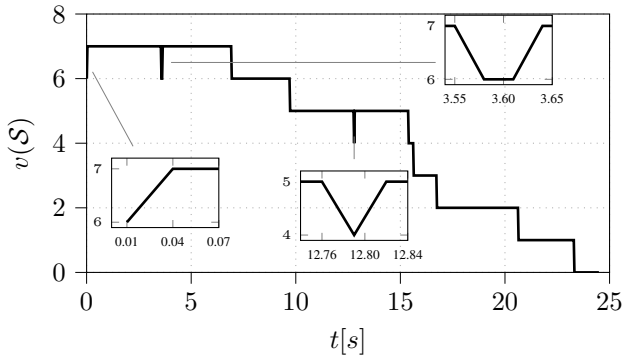
The simulation results are shown in Fig. 4 and Fig. 5. The snapshots of the game process at time instances $t = 0\,s$, $t = 5\,s$, and $t = 13.5\,s$ are illustrated in Fig. 4(a), Fig. 4(b), and Fig. 4(c), respectively. The payoff of the assignment obtained through the auction algorithm is shown in Fig. 5. Since each round of the auction algorithm lasts for $3\tau = 0.03\,s$, the assignment is updated at intervals of $0.03\,s$. As described in Section V, the assignment $\mathcal{S}^{(n)}$ can be determined after the market-clearing phase. Thus, we also update the payoff of the assignment after each market-clearing phase. For example, the payoff $v(\mathcal{S}^{(n)}) = |\mathcal{S}^{(n)}|$ is calculated at $t = \tau + 3\tau(n-1)$.

As depicted in Fig. 5(b), the distributed auction algorithm terminated in 2 rounds for the initial task assignment problem. After the first round, the payoff was $|\mathcal{S}^{(1)}| = 6$. After the second round, the auction algorithm successfully found an optimal assignment with a payoff of 7. Out of the seven assignment pairs, six involved a single robot, while one pair involved two robots. This particular pair is highlighted by the dashed box in Fig. 4(b). Throughout the game, the set $\mathcal{M}$ changed whenever an evader was captured. When such a capture happened, the existing feasible pairs involving the captured evader were removed from $\mathcal{M}$, resulting in a downward jump in the payoff $|\mathcal{S}|$, as shown in Fig. 5(b). Robots who successfully captured an evader were then eligible to bid for another task. Consequently, the assignment's payoff could increase following a downward jump. This happened at $3.64\,s$ and $12.82\,s$, as observed in the enlarged figure in Fig. 5(b). At $t = 3.58\,s$, a decrease in payoff occurred due to the capture of the evader represented by the hollow triangles in Fig. 4(b). Subsequently, the auction algorithm found another optimal solution with a payoff of 7 after two rounds, resulting in an upward jump in the payoff curve. Similarly, at $t = 12.82\,s$, there was another upward jump when a new assignment pair involving two robots (the pair highlighted by the dashed box in Fig. 4(c)) was added to the assignment.

As shown in Fig. 5, the assignment obtained through the distributed auction algorithm had an optimal payoff almost all the time, except for short periods following the changes in $\mathcal{M}$. In this simple game scenario, the auction algorithm can re-find an optimal assignment within two rounds after $\mathcal{M}$ changed. For more complex situations, the algorithm may require more



(a) Optimal payoff of Γ_0 and optimal payoff when only considering the pairs involving one robot.



(b) Payoff achieved by the auction algorithm.

Fig. 5: Optimal payoff and the payoff achieved by the distributed auction algorithm. The enlarged graphs in b provide a clearer depiction of the changes in the payoff at approximately $0.04\,s$, $3.60\,s$ and $12.80\,s$.

time before termination. However, as discussed in Section VI, if the interval between the two changes in set $\mathcal{M}$ is not too small, the auction algorithm can still terminate. Furthermore, the auction algorithm can provide feasible solutions before termination, and these feasible solutions will gradually improve until the algorithm terminates. Thus, the proposed auction algorithm with the improvements in Section VI is suitable for distributed solving of time-varying task assignment problems with robot coalitions. In addition, Fig. 5 also demonstrates that considering assignments involving two robots leads to better payoffs compared to considering only assignments involving one robot. This observation confirms the importance and effectiveness of the proposed auction algorithm.

VIII. Conclusion

This paper addressed the task assignment problem with robot coalitions, arising from practical scenarios such as multiplayer reach-avoid games. This task assignment problem is NP-hard and may vary over time. A distributed auction algorithm was proposed which can find an approximate solution to the task assignment problem with guaranteed performance in a finite number of iterations. The performance of the algorithm was evaluated through numerical simulations. The algorithm demonstrated satisfying average payoff in the conducted tests. Future work will focus on extending the algorithm to handle asynchronous communication and general communication topologies.

References

[1] B. Shirani, M. Najafi, and I. Izadi, "Cooperative load transportation using multiple uavs," *Aerospace Science and Technology*, vol. 84, pp. 158–169, 2019.

[2] D. Weyns, N. Boucké, and T. Holvoet, "Gradient field-based task assignment in an agv transportation system," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, p. 842–849, Association for Computing Machinery, 2006.

[3] K. Vivaldini, L. F. Rocha, N. J. Martarelli, M. Becker, and A. P. Moreira, "Integrated tasks assignment and routing for the estimation of the optimal number of agvs," *The International Journal of Advanced Manufacturing Technology*, vol. 82, pp. 719–736, 2016.

[4] X. Bai, Y. Ye, B. Zhang, and S. S. Ge, "Efficient package delivery task assignment for truck and high capacity drone," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 13422–13435, 2023.

[5] L. C. Batista da Silva, R. M. Bernardo, H. A. de Oliveira, and P. F. F. Rosa, "Multi-uav agent-based coordination for persistent surveillance with dynamic priorities," in *2017 International Conference on Military Technologies (ICMT)*, pp. 765–771, 2017.

[6] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing.," in *Robotics: Science and Systems*, vol. 5, pp. 343–350, Rome, Italy, 2005.

[7] X. Bai, M. Cao, W. Yan, and S. S. Ge, "Efficient routing for precedence-constrained package delivery for heterogeneous vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 248–260, 2020.

[8] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[9] C. Bererton, G. J. Gordon, and S. Thrun, "Auction mechanism design for multi-robot coordination," in *Advances in Neural Information Processing Systems* (S. Thrun, L. Saul, and B. Schölkopf, eds.), vol. 16, MIT Press, 2003.

[10] M. Chen, Z. Zhou, and C. J. Tomlin, "Multiplayer reach-avoid games via low dimensional solutions and maximum matching," in *2014 American Control Conference*, pp. 1444–1449, 2014.

[11] M. Chen, Z. Zhou, and C. J. Tomlin, "Multiplayer reach-avoid games via pairwise outcomes," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1451–1457, 2017.

[12] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[13] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[14] S. Giordani, M. Lujak, and F. Martinelli, "A distributed algorithm for the multi-robot task allocation problem," in *Trends in Applied Intelligent Systems*, (Berlin, Heidelberg), pp. 721–730, Springer Berlin Heidelberg, 2010.

[15] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the hungarian method for multirobot assignment," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 932–947, 2017.

[16] E. Garcia, D. W. Casbeer, A. Von Moll, and M. Pachter, "Multiple pursuer multiple evader differential games," *IEEE Transactions on Automatic Control*, vol. 66, no. 5, pp. 2345–2350, 2020.

[17] J. Wang, Z. Zhou, X. Jin, S. Mao, and Y. Tang, "Matching-based capture-the-flag games for multi-agent systems," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, 2023.

[18] R. Yan, X. Duan, Z. Shi, Y. Zhong, and F. Bullo, "Matching-based capture strategies for 3d heterogeneous multiplayer reach-avoid differential games," *Automatica*, vol. 140, p. 110207, 2022.

[19] R. Yan, Z. Shi, and Y. Zhong, "Task assignment for multiplayer reach–avoid games in convex domains via analytical barriers," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 107–124, 2020.

[20] R. Deng, W. Zhang, R. Yan, Z. Shi, and Y. Zhong, "Multiple-pursuer single-evader reach-avoid games in constant flow fields," *IEEE Transactions on Automatic Control*, vol. 69, no. 3, pp. 1789–1795, 2024.

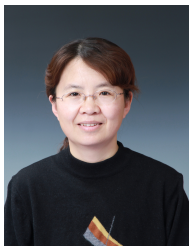
[21] E. Hazan, S. Safra, and O. Schwartz, "On the complexity of approximating k-set packing," *Computational Complexity*, vol. 15, no. 1, pp. 20–39, 2006.

[22] B. Chandra and M. M. Halldórsson, "Greedy local improvement and weighted set packing approximation," *Journal of Algorithms*, vol. 39, no. 2, pp. 223–240, 2001.
[23] M. Cygan, "Improved approximation for 3-dimensional matching via bounded pathwidth local search," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pp. 509–518, 2013.
[24] T. Thiery and J. Ward, "An improved approximation for maximum weighted k-set packing," in *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1138–1162, SIAM, 2023.
[25] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, vol. 14, no. 1, pp. 105–123, 1988.
[26] D. P. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, vol. 1, pp. 7–66, 1992.
[27] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A distributed auction algorithm for the assignment problem," in *2008 47th IEEE Conference on Decision and Control*, pp. 1212–1217, IEEE, 2008.
[28] L. Luo, N. Chakraborty, and K. Sycara, "Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 19–30, 2015.
[29] X. Bai, A. Fielbaum, M. Kronmüller, L. Knoedler, and J. Alonso-Mora, "Group-based distributed auction algorithms for multi-robot task assignment," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1292–1303, 2023.
[30] J. Fang, Z. Zhang, and R. V. Cowlagi, "Decentralized route-planning for multi-vehicle teams to satisfy a subclass of linear temporal logic specifications," *Automatica*, vol. 140, p. 110228, 2022.
[31] A. Farinelli, L. Iocchi, and D. Nardi, "Distributed on-line dynamic task assignment for multi-robot patrolling," *Autonomous Robots*, vol. 41, pp. 1321–1345, 2017.
[32] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *2008 IEEE International Conference on Robotics and Automation*, pp. 128–133, 2008.
[33] X. Bai, W. Yan, and S. S. Ge, "Distributed task assignment for multiple robots under limited communication range," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 7, pp. 4259–4271, 2022.
[34] G. Arslan, J. R. Marden, and J. S. Shamma, "Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 584–596, 2007.
[35] E. Bakolas and Y. Lee, "Decentralized game-theoretic control for dynamic task allocation problems for multi-agent systems," in *2021 American Control Conference (ACC)*, pp. 3228–3233, 2021.
[36] S. Velhal, S. Sundaram, and N. Sundararajan, "A decentralized multirobot spatiotemporal multitask assignment approach for perimeter defense," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3085–3096, 2022.
[37] O. Shorinwa, R. N. Haksar, P. Washington, and M. Schwager, "Distributed multirobot task assignment via consensus admm," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1781–1800, 2023.
[38] E. S. Lee, L. Zhou, A. Ribeiro, and V. Kumar, "Graph neural networks for decentralized multi-agent perimeter defense," *Frontiers in Control Engineering*, vol. 4, p. 1104745, 2023.
[39] K. Hirayama, "A new approach to distributed task assignment using lagrangian decomposition and distributed constraint satisfaction," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, p. 660, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.
[40] M. P. Fanti, A. M. Mangini, and W. Ukovich, "A quantized consensus algorithm for distributed task assignment," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 2040–2045, 2012.
[41] A. Testa, A. Rucco, and G. Notarstefano, "Distributed mixed-integer linear programming via cut generation and constraint exchange," *IEEE Transactions on Automatic Control*, vol. 65, no. 4, pp. 1456–1467, 2020.
[42] A. Testa and G. Notarstefano, "Generalized assignment for multi-robot systems via distributed branch-and-price," *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1990–2001, 2022.
[43] A. Falsone, K. Margellos, and M. Prandini, "A distributed iterative algorithm for multi-agent milps: Finite-time feasibility and performance characterization," *IEEE Control Systems Letters*, vol. 2, no. 4, pp. 563–568, 2018.
[44] A. Camisa and G. Notarstefano, "Primal decomposition and constraint generation for asynchronous distributed mixed-integer linear programming," in *2019 18th European Control Conference (ECC)*, pp. 77–82, 2019.
[45] A. Camisa, A. Testa, and G. Notarstefano, "Multi-robot pickup and delivery via distributed resource allocation," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1106–1118, 2023.
[46] M. Chamanbaz, G. Notarstefano, F. Sasso, and R. Bouffanais, "Randomized constraints consensus for distributed robust mixed-integer programming," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 1, pp. 295–306, 2021.
[47] A. Von Moll, E. Garcia, D. Casbeer, M. Suresh, and S. C. Swar, "Multiple-pursuer, single-evader border defense differential game," *Journal of Aerospace Information Systems*, vol. 17, no. 8, pp. 407–416, 2020.
[48] R. Yan, Z. Shi, and Y. Zhong, "Reach-avoid games with two defenders and one attacker: An analytical approach," *IEEE Transactions on Cybernetics*, vol. 49, no. 3, pp. 1035–1046, 2019.
[49] Y. Sung, A. K. Budhiraja, R. K. Williams, and P. Tokekar, "Distributed assignment with limited communication for multi-robot multi-target tracking," *Autonomous Robots*, vol. 44, no. 1, pp. 57–73, 2020.
[50] N. A. Lynch, *Distributed algorithms*. Elsevier, 1996.
[51] Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

Ruiliang Deng received the Bachelor and M.Sc. degrees from Tsinghua University, Beijing, China, in 2016 and 2019, respectively. From 2020, he is pursuing the PhD degree in Tsinghua University. His research interests include game theory, differential game and reinforcement learning.

Rui Yan received the B.E. degree in automatic control from Beihang University in 2015, and the Ph.D. degree in automatic control from Tsinghua University in 2021. He was a visiting student with the University of California, Santa Barbara from 2019 to 2020. From 2021 to 2024, he was a postdoctoral fellow with the Department of Computer Science, University of Oxford. Since 2024, he has been with the School of Artificial Intelligence, Beihang University, where he is currently an Associate Professor. His research interests focus on game theory, reach-avoid differential games, neuro-symbolic games, and multi-agent reinforcement learning.

Peinan Huang received the B.E. degree in automatic control from Shanghai Jiaotong University, Shanghai, China in 2022. He is pursuing Master's degree in Automation at Tsinghua University. His research interests include cooperative planning, formation control and reinforcement learning.



Zongying Shi (M'12) received the B.E. degree in control theory and application, the M.E. and Ph.D. degrees in control engineering from Tsinghua University, Beijing, China in 1992, 1994, and 2005, respectively. Since 1994, she has been with the Department of Automation, Tsinghua University, where she is currently an Associate Professor. Her research interests include multiagent coordination control, simultaneous localization and mapping, and robust control for robots.

Yisheng Zhong received the B.E. degree in control engineering from the Harbin Institute of Technology, Harbin, China, in 1982, the M.E. degree in electronic engineering from the University of Electro-Communications, Tokyo, Japan, in 1985, and the Ph.D. degree in electrical engineering from Hokkaido University, Sapporo, Japan, in 1988. He was a Postdoctorate Scholar with Tsinghua University, Beijing, China, from 1989 to 1990, and since 1991, he has been with the Department of Automation, Tsinghua University, where he is currently a Professor and also with the Tsinghua National Laboratory for Information Science and Technology. His research interests include robust control, nonlinear control, and differential games.