Consensus-Based Decentralized Task Allocation for Multi-Agent Systems and Simultaneous Multi-Agent Tasks

Shengli Wang, Youjiang Liu, *Member, IEEE*, Yongtao Qiu, and Jie Zhou

***Abstract*—This letter proposes a novel consensus-based timetable algorithm (CBTA) to solve the decentralized simultaneous multi-agent task allocation problem. Due to the limited capability of each agent, multiple agents may be required to perform a task simultaneously. A key challenge is how to meet the requirements and minimize the average start time of all tasks. The proposed CBTA aims to minimize the start time of each task to minimize the average start time of all tasks indirectly, it iterates between a timetable construction phase and a consensus phase. New tasks are included in the timetable of each agent by comparing the estimated start time of tasks placed by its own and other agents during the timetable construction phase. Then in the consensus phase, agents share their timetables with a communication network, and conflicts among their timetables are eliminated according to a consensus rule. Extensive simulation results show that the average start time of tasks of the proposed CBTA is nearly the same as the consensus-based bundle algorithm (CBBA) when performing single-agent tasks, and it is much less than the consensus-based grouping algorithm (CBGA) when performing multi-agent tasks with various communication network topologies.**

***Index Terms*—Decentralized algorithm, multi-agent system, task allocation and simultaneous multi-agent task.**

I. Introduction

MULTI-AGENT systems (MASs) have been widely applied in many civilization and military fields during the past few decades, typical applications include target tracking [1], intelligent transportation [2], [3] and manufacturing [4], [5], and search and rescue [6]. It is necessary for a fleet of heterogeneous agents to cooperate with each other to perform complicated tasks more efficiently and effectively. A key challenge in developing and deploying such cooperative MASs in real-world applications is task allocation, which aims to coordinate the agents to perform tasks to optimize one or more global objectives [7], [8].

Manuscript received 14 July 2022; accepted 25 October 2022. Date of publication 7 November 2022; date of current version 17 November 2022. This letter was recommended for publication by Associate Editor D. Saldaña and Editor M. Ani Hsieh upon evaluation of the reviewers' comments. *(Corresponding author: Youjiang Liu.)*

Shengli Wang is with the Department of Engineering Physics, Tsinghua University, Beijing 100084, China, and also with the Institute of Electronic Engineering, China Academy of Engineering Physics, Mianyang 621900, China (e-mail: wsl528300@163.com).

Youjiang Liu, Yongtao Qiu, and Jie Zhou are with the Institute of Electronic Engineering, China Academy of Engineering Physics, Mianyang 621000, China (e-mail: liuyj04@163.com; qiuyt685@163.com; zj_iee@163.com).

Digital Object Identifier 10.1109/LRA.2022.3220155

The existing MASs can be categorized into centralized and decentralized ones according to their team-organizational paradigms [9]. In centralized systems, the information explored by each agent, i.e., situation awareness (SA), is first collected by a central server, then an overall task allocation plan of the entire fleet is generated by the server [10], [11], [12], [13]. Optimal or near-optimal task allocation plans can be generated with centralized algorithms in such systems once globally consistent SA is achieved. However, the server in centralized systems may suffer from heavy computation burdens to generate an overall plan for the entire fleet. Besides, the mission range of the fleet is limited in such systems since all agents are required to communicate with the server, and the failure of a single agent may result in the failure of the entire system.

Therefore, decentralized systems and algorithms are developed to increase the mission range and avoid a single point of failure. Auction methods have been shown to be effective in solving the single-task allocation problem [14], [15], [16], in which an agent can be assigned a single task. In auction methods, each agent places bids on tasks individually. An auctioneer is used to collect the bids of all tasks placed by all agents and determine the winner of each task. The winner agent of a task is the one who places the highest bid on the task. The auctioneer is removed by using a message passing mechanism by some researchers so that such algorithms could be fully distributed [17]. For multi-task allocation problems in which each agent can be assigned a sequence of tasks, combinatorial auction methods were proposed [18], [19], [20]. Instead of placing bids on a single task, agents in such methods place bids on sequences of tasks. Such combinatorial auction algorithms are inefficient since the number of all possible task sequences exponentially increases as the number of tasks increases.

To solve the multi-task allocation problem efficiently, a consensus-based bundle algorithm (CBBA) was proposed [21]. CBBA iterates between a bundle construction phase and a consensus phase until no changes are made during the above two phases. It has been proven that CBBA can guarantee convergence and 50% optimality under the diminishing marginal gain (DMG) assumption. Several modifications have been made based on CBBA to extend the application scenarios and handle the dynamic uncertainties [22], [23]. Based on CBBA, a heuristic performance impact (PI) algorithm was proposed to deal with the time constraints of tasks [24]. Numerical simulations show that PI performs better than CBBA when applied to time-critical

scenarios. Based on PI algorithm, various rescheduling methods have been proposed to further decrease the average start time of tasks [25] or maximize the total number of task allocations [26]. However, much more iterations are required by the algorithms to converge to better results due to the rescheduling. Besides, all these algorithms can only be applied to single-agent task allocation problems in which only a single agent is required by each task.

In some real-world applications, multiple agents may be required by each task due to the limited capability of each agent. A consensus-based grouping algorithm (CBGA) was proposed to solve the multi-agent task allocation problem [27], [28]. When the agents for a task are required to perform the task simultaneously, the results of CBGA may be infeasible. The result of CBGA is said to be feasible if the agents can perform tasks according to the task allocation plan generated by CBGA without any conflicts, and infeasible otherwise. With CBGA, the agents for a task may never reach the task simultaneously in some scenarios. For example, assuming that two agents are required by tasks 1 and 2, agent 1 performs tasks 2 and 1 sequentially, while agent 1 performs tasks 1 and 2 sequentially with CBGA. When performing tasks, agent 1 is waiting for agent 2 at the position of task 2, while agent 2 is waiting for agent 1 at the position of task 1. Therefore, neither tasks 1 nor 2 can be successfully executed.

In this letter, a consensus-based timetable algorithm (CBTA) is proposed to solve the simultaneous multi-agent task allocation problem. The proposed CBTA aims to meet the requirements and minimize the average start time of tasks, and it iterates between a timetable construction phase and a consensus phase. The advantages of the proposed CBTA are as follows. First, the proposed approach can generate feasible solutions for all simultaneous multi-agent task allocation problems, while the state-of-the-art CBGA cannot. Second, the average start time of tasks of the proposed CBTA is much less than CBGA. Extensive simulations demonstrate the outstanding performance of the proposed algorithm compared with the state-of-the-art CBBA and CBGA.

The rest of this letter is organized as follows. In section II, the simultaneous multi-agent task allocation problem is first described and mathematically formulated. Then the proposed CBTA is presented in section III. Extensive simulations are presented and analyzed in section IV to demonstrate the outstanding performance of the proposed CBTA. Finally, section V concludes this letter.

II. Problem Formulation

Consider a fleet of N_a agents used to perform a set of N_t tasks. The agents are required to arrive at the location of a task first before executing the task. The agents are coordinated in a decentralized manner, and an overall task allocation plan is generated with a communication network. Due to the limited capability of each agent, multiple agents may be required to perform a task simultaneously. For example, if the agents are unmanned vehicles, and the task is to locate the position of a target by measuring the time difference of arrival (TDOA) in 2-dimensional space, at least three vehicles are required to receive the signal of the target simultaneously [29]. Each agent can perform multiple tasks sequentially, and it is not required to return to its initial position after finishing all its tasks. A key challenge is how to find an optimal allocation to meet the requirements and minimize the average start time of all tasks.

TABLE I
Symbol Definition

Symbol	Definition
$\mathcal{A} = \{1, 2, \ldots, N_a\}$	A set of N_a agents
$\mathcal{T} = \{1, 2, \ldots, N_t\}$	A set of N_t tasks
L_t	The maximum number of tasks that can be assigned to each agent
R_k	Number of agents required by task k
δ_k	Time required to execute task k
$\mathbf{T}_i$	The timetable of agent i, T_{ijk} denotes the estimated start time when agent j is going to perform task k
$\mathbf{a}_{ik}$	Agents for task k in agent i's estimation
τ_{ik}	Predicted start time of task k in agent i's estimation
$\mathbf{b}_{ij}$	Tasks being performed by agent j in agent i's estimation
$\mathbf{p}_{ij}$	The sequence that agent j performs its tasks in agent i's estimation
$\mathbf{b}_i$	The tasks being performed by agent i in agent i's estimation
$\mathbf{p}_i$	The sequence that agent i performs its tasks in agent i's estimation
$\lvert \cdot \rvert$	The cardinality of a set
$\mathbf{p}_i \oplus_l k$	The task path after inserting task k before the lth element of $\mathbf{p}_i$
$\mathbf{T}_i \oplus_l k$	The timetable of agent i after inserting task k before the lth element of $\mathbf{p}_i$
$c_{ik}(\mathbf{T}_i)$	The marginal cost of inserting task k in the task path of agent i
$\mathbf{G}$	The communication network topology of the agent fleet
$\mathbf{s}_i$	The communication timestamp list of agent i, which records the latest time when agent i communicates with other agents
r_{ik}	The number that task k is removed from the timetable of agent i
γ	The maximum number that a task can be removed from the task list of an agent

In order to formulate the task allocation problem mathematically, let $\mathcal{A} = \{1, 2, \ldots, N_a\}$ be the fleet of agents and $\mathcal{T} = \{1, 2, \ldots, N_t\}$ be the set of tasks being performed. The key symbols used hereafter are listed in Table I. A maximum of L_t tasks can be assigned to each agent due to its limited capability. R_k agents are required by task k, and a duration time δ_k is required for the agents to execute the task, i.e., the time period from when the agents for the task start to execute it to when it is successfully executed. Therefore, the total number of agents should be bigger than the maximum requirement of tasks, i.e., $N_a \geqslant \max_{k=1}^{N_t} R_k$, so that the agents have enough capabilities to execute each task. Besides, the total capabilities of all agents should be enough for all tasks, which represents that the total number of tasks that can be assigned to all agents should be more than the total requirements of all tasks, i.e., $N_a L_t \geqslant \sum_{k=1}^{N_t} R_k$.

The agents for a task are required to execute it simultaneously, but they may arrive at the task location at different times due to their different positions and cruising speeds. Therefore, the agents for a task arriving earlier are required to wait for those who arrive later until the latest one arrives, and the real start time of a task is when the latest agent for the task arrives. The start time of a task performed by an agent is dependent on the

start time of other agents in the simultaneous multi-agent task allocation problem. Therefore, the estimated start time of tasks placed by all agents should be aware by each agent to determine the real start time of each task.

A timetable $\mathbf{T}_i$ is kept on agent i to record the estimated start time of all tasks placed by all agents, which is a matrix sized $N_a \times N_t$. $T_{ijk} > 0$ represents agent i estimates that agent j is going to perform task k at T_{ijk}, and $T_{ijk} = 0$ if agent i estimates that agent j is not going to perform task k. Assuming that the timetables of all agents are free of conflicts, i.e., the timetables of different agents are the same, and all tasks can be successfully executed according to the timetables. Let $\mathbf{a}_{ik}$ be the agent squad of task k in the estimation of agent i, which contains the agents for the task, then $\mathbf{a}_{ik} = \{j \in \mathcal{A} | T_{ijk} > 0\}$, and the real start time of task k is $\tau_{ik} = \max_{j=1}^{N_a} T_{ijk}$. For each agent j, let $\mathbf{b}_{ij}$ be the tasks being performed by the agent in the estimation of agent i, then $\mathbf{b}_{ij} = \{k \in \mathcal{T} | T_{ijk} > 0\}$. Let $\mathbf{p}_{ij}$ be the task path of agent j in the estimation of agent i, which is an ordered sequence of tasks, its lth element $p_{ijl} = k$ if agent i estimates that agent j is going to perform task k at the lth point according to $\mathbf{p}_{ij}$. Given the timetable $\mathbf{T}_i$, the real start time of tasks in $\mathbf{b}_{ij}$ can be easily found, and the task path $\mathbf{p}_{ij}$ can be found by sorting their real start time. In order to simplify the notations, let $\mathbf{b}_i$ and $\mathbf{p}_i$ be the tasks and task path of agent i in the estimation of agent i in the following of this letter, i.e., $\mathbf{b}_i = \mathbf{b}_{ii}$ and $\mathbf{p}_i = \mathbf{p}_{ii}$.

Given the above mathematical formulations, the simultaneous multi-agent task allocation problem can be formulated as the following optimization problem.

$$\min \left\{ \frac{1}{N_t} \sum_{k=1}^{N_t} \max_{j=1}^{N_a} T_{ijk} \right\} \tag{1}$$

s.t.

$$\mathbf{T}_i = \mathbf{T}_j, \forall (i,j) \in \mathcal{A}^2, i \neq j \tag{2}$$

$$\sum_{j=1}^{N_a} h(T_{ijk} > 0) = R_k, \forall (i,k) \in \mathcal{A} \times \mathcal{T} \tag{3}$$

$$\sum_{k=1}^{N_t} h(T_{ijk} > 0) \leqslant L_t, \forall (i,j) \in \mathcal{A}^2 \tag{4}$$

$$\sum_{j=1}^{N_a} \sum_{k=1}^{N_t} h(T_{ijk} > 0) = \sum_{k=1}^{N_t} R_k, \forall i \in \mathcal{A} \tag{5}$$

The objective of agent i in (1) is to minimize the average start time of all tasks. Note that the objectives of different agents may be different since they are coordinated in a distributed manner. Constraint (2) denotes that the timetables of different agents should be the same so that the assignments of different agents can be free of conflicts. Note that once constraint (2) is met, the objective in (1) is the same for different agents. In (3), $h(\cdot)$ equals 1 if the condition is true and 0 otherwise. Equation (3) represents that the number of agents for task k should meet its requirement, while (4) represents that the number of tasks assigned to agent i should be no more than its maximum capability. Equation (5) represents that the total assignments of all agents should meet the total requirements of all tasks.

III. Consensus-Based Timetable Algorithm

A. Time Table Construction Phase

Given the timetable $\mathbf{T}_i$, the agents for task k can be easily found as $\mathbf{a}_{ik} = \{j \in \mathcal{A} | T_{ijk} > 0\}$ if $\mathbf{T}_i$ is free of conflicts. However, the number of agents in $\mathbf{a}_{ik}$ may be unequal to the requirement of task k during the optimization process, i.e., $|\mathbf{a}_{ik}| \neq R_k$, where $|\cdot|$ denotes the cardinality of the set. When the number of agents in $\mathbf{a}_{ik}$ is not bigger than the requirement of task k, i.e., $|\mathbf{a}_{ik}| \leqslant R_k$, all the agents in $\mathbf{a}_{ik}$ are used to execute the task in the estimation of agent i. The number of agents in $\mathbf{a}_{ik}$ may be more than the requirements of task k sometimes, i.e., $|\mathbf{a}_{ik}| > R_k$. Since the agents for each task should perform it simultaneously, the real start time of each task is when the latest agent for the task arrives. In order to minimize the real start time of task k, the agent in $\mathbf{a}_{ik}$ that yields the maximum start time is removed one by one until constraint (3) is met, i.e., the agents in $\mathbf{a}_{ik}$ equals the requirement of task k. Note that if more than one agent yields the maximum start time, the agents with smaller IDs are removed first.

Given the agent squad $\mathbf{a}_{ik}$, the temporary real start time of task k in the estimation of agent i is considered as

$$\tau_{ik} = \begin{cases} T_{iik}, & \text{if } T_{iik} > 0 \text{ and } i \notin \mathbf{a}_{ik} \\ \max_{j \in \mathbf{a}_{ik}} T_{ijk}, & \text{otherwise} \end{cases} \tag{6}$$

If agent i estimates it is used to perform task k, but it is not in the agent squad of task k, i.e., $T_{iik} > 0$ and $i \notin \mathbf{a}_{ik}$, the real start time of task k is considered as the estimated time when agent i starts task k. In such case, $T_{iik} > \tau_{ik}$ always holds, and it will be set to 0 in the following consensus phase. Otherwise, if agent i thinks it is not used to perform task k or it is in the agent squad of task k, i.e., $T_{iik} = 0$ or $i \in \mathbf{a}_{ik}$, the real start time of task k is considered as the time when the latest agent in $\mathbf{a}_{ik}$ arrives. Note that, (6) represents the temporary start time of tasks estimated by agent i, and may violate some of the constraints enforced on the final allocations.

Given the timetable $\mathbf{T}_i$ and the real start time of each task, the total cost in the estimation of agent i is defined as

$$S_i(\mathbf{T}_i) = \sum_{k=1}^{N_t} \{(R_k - |\mathbf{a}_{ik}|) \cdot U + \tau_{ik}\} \tag{7}$$

where U is a constant value that is big enough to be bigger than the total start time of all tasks, i.e., $U > \sum_{k=1}^{N_t} \tau_{ik}, \forall i \in \mathcal{A}$, thus the agents are encouraged to make proposals for the tasks whose requirements have not been met yet. Once the requirements of all tasks are met, then $\sum_{k=1}^{N_t} (R_k - |\mathbf{a}_{ik}|) \cdot U = 0$, the cost in (7) represents the total start time of all tasks. Therefore, to minimize the cost in (7), not only the requirements of all tasks should be met, but the total start time of tasks should be minimized.

The tasks being performed by agent i are $\mathbf{b}_i = \{k \in \mathcal{T} | T_{iik} > 0\}$, and the task path $\mathbf{p}_i$ can be found by sorting the real start time of these tasks. A removal list $\mathbf{r}_i$ is kept on agent i to record the number that each task was removed from $\mathbf{T}_i$, and a maximum number γ that a task can be removed from the timetable of an agent is set to guarantee convergence. Given the task path $\mathbf{p}_i$, a candidate task can be inserted before any element or at the end of $\mathbf{p}_i$. A task is said to be a candidate of agent i if it has not been

Algorithm 1: Timetable Construction Phase of Agent i.

while $\sum_{k=1}^{N_t} h(T_{iik} > 0) \leqslant L_t$ **do**
  Computed $c_{ik}, \forall k \in \mathcal{T} \setminus \mathbf{p}_i$ according to (9)
  $g = \max_{k \leqslant N_t} c_{ik}$
  if $g \geqslant 0$ **then**
    $q = \arg\max_{k \leqslant N_t} c_{ik}$
    $n = \arg\max_{l \leqslant |\mathbf{p}_i|+1} \{S_i(\mathbf{T}_i) - S_i(\mathbf{T}_i \oplus_l q)\}$
    Find $\mathbf{a}_{i,q}$ according to $\mathbf{T}_i \oplus_n q$
    if $i \in \mathbf{a}_{i,q}$ **then**
      Update the timetable, $\mathbf{T}_i \leftarrow \mathbf{T}_i \oplus_n q$
      Update the task path, $\mathbf{p}_i \leftarrow \mathbf{p}_i \oplus_n q$
    else
      break
    end if
  else
    break
  end if
end while

in $\mathbf{p}_i$ yet and the number of removals is less than the limitation, i.e., $k \notin \mathbf{p}_i$ and $r_{ik} < \gamma$. Inserting task k at the lth position of $\mathbf{p}_i$, denoted as $\mathbf{p}_i \oplus_l k$, represents that task k is inserted before the lth element of $\mathbf{p}_i$ if $l \leqslant |\mathbf{p}_i|$ or at the end of $\mathbf{p}_i$ if $l = |\mathbf{p}_i| + 1$. After inserting task k at the lth position of $\mathbf{p}_i$, i.e., $\mathbf{p}_i \leftarrow \mathbf{p}_i \oplus_l k$, the estimated start time of tasks in $\mathbf{T}_i \oplus_l k$ is required to be updated according to (8).

$$T_{i,i,p_{ik}} = \begin{cases} \frac{d^{(\mathrm{a})}_{i,p_{ik}}}{v_i}, & \text{if } k = 1 \\ \tau_{i,p_{i,k-1}} + \delta_{p_{i,k-1}} + \frac{d^{(\mathrm{t})}_{p_{i,k-1},p_{ik}}}{v_i}, & \text{otherwise} \end{cases} \tag{8}$$

where $d^{(\mathrm{a})}_{i,p_{ik}}$ denotes the distance between agent i and task p_{ik}, v_i denotes the cruising speed of agent i, $\delta_{p_{i,k-1}}$ denotes the duration time of performing task $p_{i,k-1}$. $d^{(\mathrm{t})}_{p_{i,k-1},p_{ik}}$ denotes the distance from the former task to the latter one according to $\mathbf{p}_i$. Equation (8) represents that the estimated start time of agent i performing the first task in $\mathbf{p}_i$ is when it arrives at the position of the task from its initial position, and the estimated start time of performing the other tasks in $\mathbf{p}_i$ is when it arrives at the positions of these tasks after finishing the former tasks.

The marginal cost of inserting task k into the task path of agent i is defined as the maximum decrease in the total cost in the estimation of agent i, which is

$$c_{ik}(\mathbf{T}_i) = \begin{cases} -U, & \text{if } T_{iik} > 0 \text{ or } r_{ik} > \gamma \\ \max_{l=1}^{|\mathbf{p}_i|+1} \{S_i(\mathbf{T}_i) - S_i(\mathbf{T}_i \oplus_l k)\}, & \text{otherwise} \end{cases} \tag{9}$$

The procedures of the timetable construction phase running on agent i are summarized as Algorithm 1. Candidate tasks are included in the timetable of agent i one by one by comparing their marginal cost while the number of tasks being performed by agent i is less than L_t. Given the timetable $\mathbf{T}_i$, the task path $\mathbf{p}_i$ is first generated by sorting the real start time of tasks being performed by agent i. Then the marginal costs of candidate tasks are computed according to (9). The proposed CBTA aims to meet the requirements and minimize the average start time of all tasks.

Algorithm 2: Consensus Phase of Agent i.

Send $\mathbf{T}_i$ and $\mathbf{s}_i$ to agent m if $G_{im} = 1$
Receive $\mathbf{T}_m$ and $\mathbf{s}_m$ from agent m if $G_{im} = 1$
for $j = 1$ to N_u **do**
  if $j = m$ **then**
    $T_{ijk} = T_{mjk}, \forall k \in \mathcal{T}$
  else
    if $j \neq i$ and $s_{mj} > s_{ij}$ **then**
      $T_{ijk} = T_{mjk}, \forall k \in \mathcal{T}$
    end if
  end if
end for
for $k = 1$ to N_t **do**
  while $\sum_{j=1}^{N_a} h(T_{ijk} > 0) > L_k$ **do**
    $q = \arg\max_{j=1}^{N_a} T_{ijk}$
    $T_{iqk} = 0$
    if $q = i$**then**
      $r_{ik} = r_{ik} + 1$
    end if
  end while
end for
Update $\mathbf{T}_i$ according to (8)

Therefore, the task that yields the maximum decrease in the total cost is first assigned to agent i.

$$g = \max_{k=1}^{N_t} c_{ik}(\mathbf{T}_i) \tag{10}$$

Let q be the task that yields the maximum marginal cost g, and n be the position where q is inserted at $\mathbf{p}_i$. If $g > 0$, the total cost in the estimation of agent i can be further decreased by inserting q into the $\mathbf{p}_i$. Note that when $g = 0$, the estimated start time of task q placed by agent i is the same as some other agents, and the requirement of q has already been met. The agents $\mathbf{a}_{i,q}$ for task q is first found according to $\mathbf{T}_i \oplus_n q$. If $i \in \mathbf{a}_{i,q}$, it is necessary to add task q to the task path of agent i. If $g < 0$ or $i \notin \mathbf{a}_{i,q}$, the total cost in the estimation of agent i cannot be further decreased by inserting any of the rest tasks in $\mathbf{p}_i$, then the timetable construction phase comes to an end.

B. Consensus Phase

The timetable construction phase runs on different agents concurrently, thus conflicts may happen among their timetables. For example, the number of agents for each task is unequal to its requirements. The consensus phase is utilized to eliminate conflicts and meet the requirements of tasks. Let $\mathbf{G}$ be the communication network topology of the agent fleet, $G_{ij} = 1$ if agent i and j can communicate with each other during each iteration and 0 otherwise. In the consensus phase, agent i sends its timetable $\mathbf{T}_i$ to other agents if a communication link between them exists. Besides, a time stamp list $\mathbf{s}_i$ is kept on agent i to record the latest time when it communicated with other agents. In the consensus phase, $\mathbf{s}_i$ is also sent to other agents to determine the information of which agent is the most up-to-date.

The procedures of the consensus phase running on agent i are summarized as Algorithm 2. When receiving the timetable

$\mathbf{T}_m$ and the time stamp list $\mathbf{s}_m$, the estimated start time of tasks of agent m in the estimation of agent i is updated according to $\mathbf{T}_m$, i.e., $T_{imk} = T_{mmk}, \forall k \in \mathcal{T}$ (Line 4-6 in Algorithm 2). For another agent j except i and m, if $s_{mj} > s_{ij}$, the information about agent j in the estimation of agent m is more up-to-date than agent i, then the information about agent j in $\mathbf{T}_i$ is updated according to $\mathbf{T}_m$ (Line 7-9 in Algorithm 2).

After updating the information of other agents according to their timetables, the agents for a task in $\mathbf{T}_i$ may be more than the requirement of the task, i.e., $\sum_{j=1}^{N_a} h(T_{ijk} > 0) > R_k$. Therefore, the timetable of each agent should be further updated. In CBBA and CBGA, a bundle is used to record the sequence that tasks are assigned to an agent. Once a task is outbid during the consensus phase, the tasks assigned to the agent later than this task are all released. However, in the proposed CBTA, multiple agents are required to perform a task simultaneously. When a task is removed from the task path of an agent, then the agents for the task may be insufficient, and changes should also be made to the assignments of other agents that are used to perform tasks cooperatively. Thus the algorithm will converge at a very low speed. Therefore, the bundle is not used in the proposed CBTA, and only the unnecessary agents for each task are removed. While the number of agents for task k in $\mathbf{T}_i$ is more than the requirement of the task, the agent that yields the maximum estimated start time is set to 0 one by one until constraint (3) is met, i.e., the number of agents for task k equals its requirement (Line 12-20 in Algorithm 2). Note that if the estimated start time placed by agent i is set to 0, then the number that task k is removed should be updated, i.e., $r_{ik} \leftarrow r_{ik} + 1$ (Line 16-18 in Algorithm 2). During the above procedures, some tasks being performed by agent i may be removed, the task path $\mathbf{p}_i$ should be recomputed, and the estimated start time of tasks in $\mathbf{p}_i$ needs to be updated according to (8) (Line 21 in Algorithm 2).

C. Convergence

For single-agent task allocation problems, CBBA is proven to guarantee convergence under the DMG assumption [21]. As for the multi-agent task allocation problem in which the agents for a task are not required to perform the task simultaneously, each task can be treated as multiple tasks with the same location, and a single agent is required by each of these tasks, thus the convergence of CBGA can also be guaranteed.

However, when the agents for each task are required to start the task simultaneously, the assignment made by each agent is dependent on the assignments of other agents. Due to the incomplete communication links in the network topologies of the agent fleet, the information in the timetable of agent i about agent j may be out-of-date sometimes. The agents change their estimated start time of a task since the real start time of the task can be further decreased in its estimation. However, the estimated start time of the task placed by an agent has already been updated when it is received by the other agents that are used to perform the task simultaneously. Therefore, the proposed approach may be trapped into infinite cycles by exchanging some tasks between some agents, and it can never converge to a feasible solution within finite iterations. To guarantee convergence, the number that an agent can remove a task is limited [26].

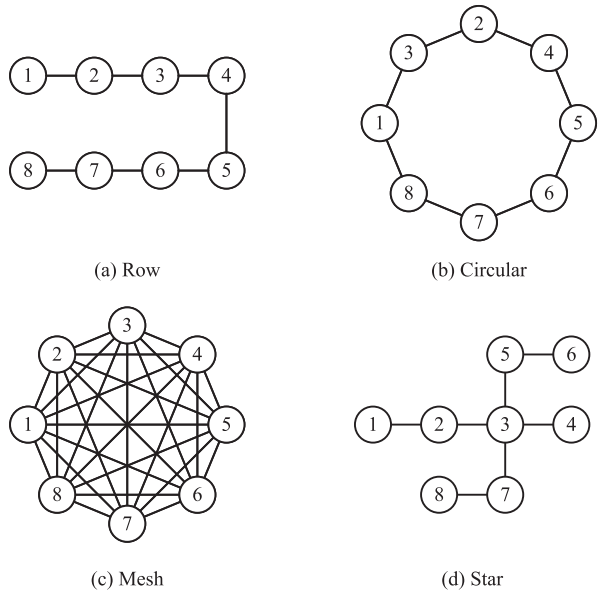


Fig. 1. Communication network topologies used in the simulations with 8 agents, two agents can communicate with each other during each iteration if they are connected in the topologies.

Once the number that a task is removed from the timetable of an agent reaches the maximum limitation, then the agent will hold its estimated start time of the task. And the fixed start time of the task is sent to other agents during the following iterations, then the other agents construct their timetable according to the fixed start time. Therefore, the agents for the task can be fixed. Then the algorithm will converge as the agents for each task are fixed recursively. If the maximum number of removals is too small, the algorithm may not converge to a feasible solution before it comes to an end, so it should be sufficiently large. However, too many iterations may be required by the algorithm to converge if the number is too large. Therefore, the number that a task can be removed by an agent should be sufficiently but not too large.

IV. Simulations and Results

A. Scenarios and Parameter Configurations

In the simulations considered in this letter, agents and tasks were uniformly generated in a 10 000 $\times$ 10 000 $\times$ 1000 m area. The number of agents required by each task was set to 1 when compared with CBBA, while it was uniformly set between 1 and 3 when compared with CBGA on multi-agent tasks. The cruising speeds of the agents were assumed to be constant and uniformly set between 20 to 40 m/s. The duration time of each task was uniformly set between 200 and 400 s. The communication network topologies illustrated in Fig. 1 were utilized in the simulations.

Since most parameters were randomly generated, 100 independent runs were conducted for each specified number of agents and tasks. It is worth pointing out that for multi-agent tasks, the results of CBGA may be infeasible, i.e., not all tasks can be executed successfully. To make comparisons, only the feasible scenarios for CBGA were considered.

TABLE II
THE AVERAGE START TIME OF TASKS, NUMBER OF ITERATIONS AND CPU TIME TAKEN BY EACH AGENT OF CBBA AND CBTA AMONG 100 RUNS WITH VARIOUS NUMBER OF AGENTS AND SINGLE-AGENT TASKS

Scenario		Start time/s		Iterations		CPU time/s	
Agents	Tasks	CBBA	CBTA	CBBA	CBTA	CBBA	CBTA
4	4	127.59	131.31	3.42	3.31	0.003	0.007
4	6	224.23	225.98	4.27	3.81	0.005	0.017
4	8	293.86	294.33	5.49	4.08	0.011	0.036
8	8	111.73	113.78	7.63	7.71	0.010	0.041
8	12	193.72	194.88	9.26	8.64	0.033	0.141
8	16	265.16	270.91	11.54	9.40	0.089	0.357
12	12	97.16	97.30	12.40	11.97	0.048	0.146
12	18	177.19	178.43	14.98	14.31	0.192	0.558
12	24	247.89	253.68	19.03	15.53	0.555	1.517

B. Comparison With CBBA on Single-Agent Tasks

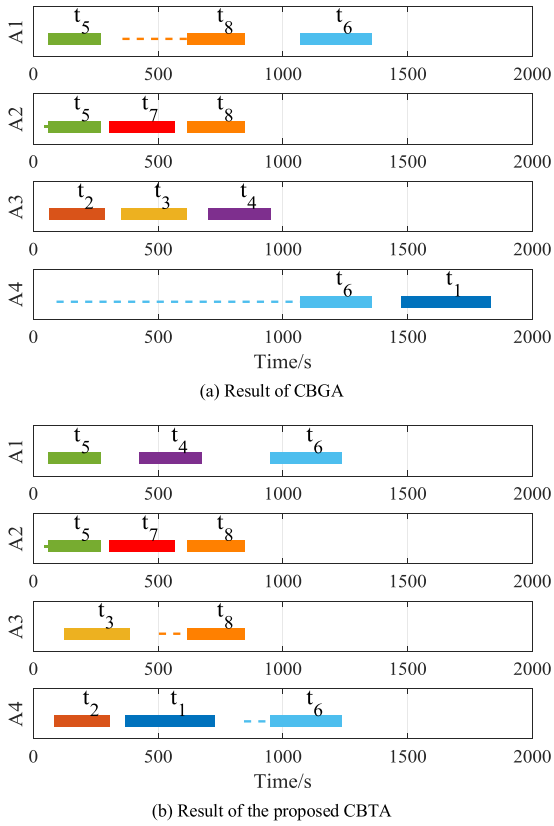
Table II shows the average start time of tasks, the number of iterations, and CPU time taken by each agent of CBBA and the proposed CBTA when performing single-agent tasks, the row network topology illustrated in Fig. 1(a) was utilized in the simulations. As shown in the table, the average start time of tasks of the proposed CBTA is a little bigger than CBBA, and the number of iterations of the proposed CBTA is always less than CBBA, while the average CPU time of the proposed CBTA is always more than CBBA.

Once a task in CBBA is outbid, the tasks assigned to the agent later than the task are all released according to the bundle in CBBA. However, with CBTA, a task is removed from the timetable of an agent only when it is unnecessary for the task. Therefore, the average start time of tasks of the proposed CBTA is a little bigger than CBBA, but CBTA requires fewer iterations to converge. During the timetable construction phase of CBTA, the task path of each agent is first generated according to its timetable, while the estimated start time of other agents is considered by each agent. Hereby, more CPU time is required by the proposed CBTA.

C. Comparison With CBGA on Multi-Agent Tasks

The proposed CBTA was further compared with CBGA on multi-agent tasks. The number of agents was set to 4, 8, and 12, while the number of tasks was set to 4, 6, ..., 16, and the row network topology was utilized. Note that CBGA cannot generate feasible solutions for most scenarios when the agents for each task are required to perform the task simultaneously. With 16 multi-agent tasks, a total of 5090, 461, and 199 runs was conducted to find 100 feasible scenarios for CBGA with 4, 8, and 12 agents respectively.

Fig. 2 depicts a sample time schedule of 4 agents and 8 multi-agent tasks of CBGA and the proposed CBTA. As shown in Fig. 2(a), the tasks requiring fewer agents are scheduled to be performed later than those requiring more agents with CBGA, which leads to more waiting time. For example, task 1 is scheduled to be performed by agent 4 after task 6 in Fig. 2(a), then it is executed much later due to the waiting time of agent 4 performing task 6. Agents in CBGA make their own decisions independently and are more likely to select the nearest tasks,



(a) Result of CBGA

(b) Result of the proposed CBTA

Fig. 2. Time schedule of 4 agents and 8 multi-agent tasks. Each horizontal colored line denotes the time period for executing the corresponding task, and the lines in the schedule of different agents with the same color represent that these agents are used to perform the task cooperatively. Each colored dashed line denotes the time period of the agent to wait for other agents. For example, in (a), A_1 and A_2 are used to perform task t_8 cooperatively, A_1 arrives at the location of t_8 earlier than A_2 since A_2 needs to finish t_7 before t_8. (a) Result of CBGA, the average start time of the tasks is 579.73 s. (b) Result of the proposed CBTA, the average start time is 366.00 s.

which leads to unnecessary time costs when the agents for a task are required to start the task simultaneously. However, with the proposed CBTA, agents make their own according to the estimated start time of other agents, which leads to an earlier start time of tasks.

Fig. 3 depicts the average start time of tasks of CBGA and the proposed CBTA. The average start time of tasks of the proposed CBTA is always less than CBGA. As the number of agents increases, the average start time of tasks decreases, while the gap between CBGA and the proposed CBTA also decreases. As the number of tasks increases, the average start time of tasks increases, while the gap between CBGA and the proposed CBTA also increases.

Fig. 4 depicts the number of iterations taken by CBGA and the proposed CBTA to converge. Let θ be the average number of tasks being performed by each agent, i.e., $\theta = \sum_{k=1}^{N_t} R_k / N_a$. When θ is small, each agent is only required to perform a few tasks on average. With CBGA, once a task is removed from the assignment of an agent, the tasks that are added to the task path of the agent later than the removed task are all released. However, with the proposed CBTA, a task is removed from the timetable of an agent only when the agent is unnecessary for the task.

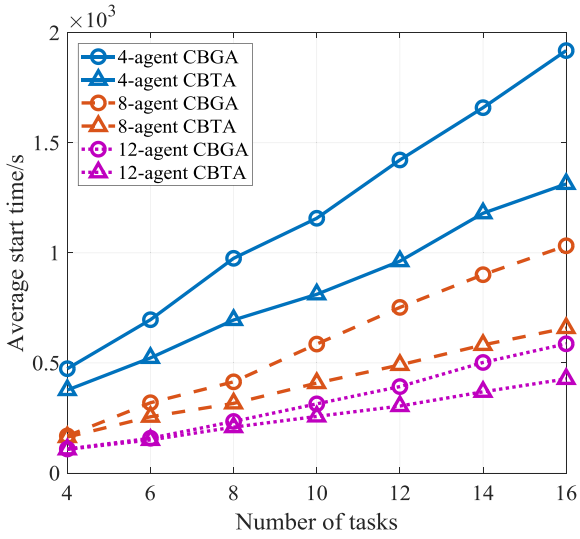


Fig. 3. The average start time of tasks of CBGA and the proposed CBTA when performing multi-agent tasks.

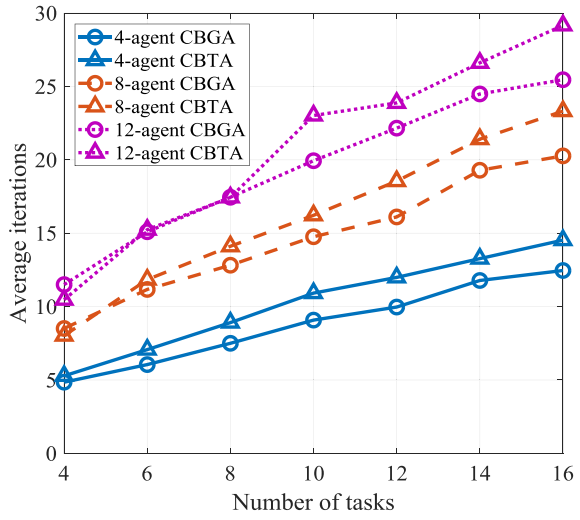


Fig. 4. The average iterations taken by CBGA and the proposed CBTA to converge when performing multi-agent tasks.

Therefore, the number of iterations taken by CBGA to converge is more than the proposed CBTA when θ is small. As the number of tasks increases, the average number of tasks being performed by each agent also increases. In the proposed CBTA, the agents for a task are required to start the task simultaneously. Then more iterations are required to determine the agents for each task. Therefore, more iterations are required to converge by the proposed CBTA when θ is big.

Fig. 5 depicts the average CPU time taken by each agent to converge of CBGA and the proposed CBTA. The average CPU time taken by each agent with the proposed CBTA is always bigger than CBGA despite the number of agents and tasks. During the timetable construction phase of CBTA, the task path of each agent is required to be first generated according to its timetable, while the estimated start time of other agents is

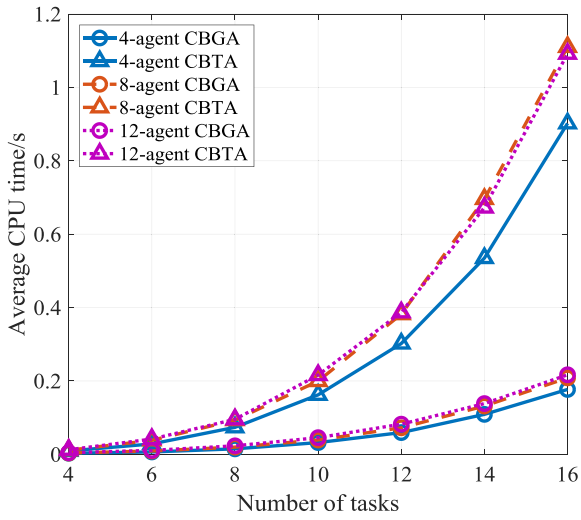


Fig. 5. The average CPU time taken by each agent of CBGA and the proposed CBTA to converge when performing multi-agent tasks.

TABLE III
THE AVERAGE START TIME OF TASKS, NUMBER OF ITERATIONS, AND CPU TIME TAKEN BY EACH AGENT OF CBGA AND THE PROPOSED CBTA AMONG 100 RUNS WITH VARIOUS NETWORK TOPOLOGIES

Scenario	Start time/s		Iterations		CPU time/s	
	CBGA	CBTA	CBGA	CBTA	CBGA	CBTA
Mesh	732.12	495.71	8.91	8.01	0.07	0.22
Row	733.36	495.46	16.55	19.12	0.08	0.37
Circular	733.29	480.17	10.68	9.98	0.07	0.25
Star	733.29	487.34	13.63	16.40	0.07	0.34

considered by each agent, which leads to more computational complexity. As the number of agents increases, the average CPU time taken by each agent increases slowly, and it is almost the same with various numbers of tasks with 8 and 12 agents. The agents are coordinated in a distributed manner, and the computational complexity of the algorithm running on each agent is mainly dependent on the number of tasks.

D. Comparison on Network Topologies

The proposed CBTA was further investigated with the four network topologies illustrated in Fig. 1, the number of agents was set to 8, while the number of tasks was set to 12.

The average start time of tasks, number of iterations, and CPU time taken by each agent of CBGA and the proposed CBTA are listed in Table III. Note that a total of 197 runs were used by CBGA to find 100 feasible scenarios for all network topologies. The average start time of tasks of the proposed CBTA is always much less than CBGA for all investigated network topologies. The number of iterations used by CBTA to converge is less than CBGA with mesh and circular topologies, while it is more than CBGA with row and star topologies. As shown in Fig. 1, the communication conditions of mesh and circular topologies are better than row and star topologies, i.e., more agents can exchange information with each other during each iteration, and the network diameter is lower. The agents for a task are required

to start the task simultaneously in the proposed CBTA, thus more information is required to be exchanged among the agents before the algorithm is converged. Therefore, more iterations are required by CBTA to converge as the communication quality of the agent fleet becomes worse. The average CPU time required by the proposed CBTA is always more than CBGA as the computational complexity of the proposed CBTA is higher.

V. Conclusion

In this letter, a decentralized simultaneous multi-agent task allocation method named CBTA has been presented. The proposed CBTA iterates between the timetable construction phase and consensus phase until no changes are made during the above two phases. New tasks are included in the timetable of each agent by comparing the estimated start time of tasks placed by its own and other agents during the timetable construction phase, while the conflicts among the timetables of different agents are eliminated in the consensus phase. Extensive simulations have been presented to demonstrate the outstanding performance of the proposed approach. Compared with CBGA, the proposed CBTA can generate feasible solutions for all scenarios, and the average start time of tasks is much less. In future work, we are interested in decreasing the computational complexity and extending the application scenarios of the proposed CBTA, and we are also interested in solving the task allocation problem in which an agent executes several tasks simultaneously.

References

[1] J. Yan, X. Guan, and F. Tan, "Target tracking and obstacle avoidance for multi-agent systems," *Int. J. Autom. Comput.*, vol. 7, no. 4, pp. 550–556, Nov. 2010.

[2] M. Dotoli, H. Zgaya, C. Russo, and S. Hammadi, "A multi-agent advanced traveler information system for optimal trip planning in a co-modal framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2397–2412, Sep. 2017.

[3] S. Satunin and E. Babkin, "A multi-agent approach to intelligent transportation systems modeling with combinatorial auctions," *Expert Syst. Appl.*, vol. 41, no. 15, pp. 6622–6633, May 2014.

[4] I. Kovalenko, D. Tilbury, and K. Barton, "The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems," *Control Eng. Pract.*, vol. 86, no. 1, pp. 105–117, Mar. 2019.

[5] B. Hu and J. Chen, "Optimal task allocation for human–machine collaborative manufacturing systems," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 1933–1940, Oct. 2017.

[6] D. S. Drew, "Multi-agent systems for search and rescue applications," *Curr. Robot. Rep.*, vol. 2, no. 2, pp. 189–200, Mar. 2021.

[7] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Inform.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[8] S. Alatartsev, S. Stellmacher, and F. Ortmeier, "Robotic task sequencing problem: A survey," *J. Intell. Robot. Syst.*, vol. 80, no. 2, pp. 279–298, 2015.

[9] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Study Comput. Intell.*, vol. 604, pp. 31–51, 2015.

[10] S. K. K. Hari, A. Nayak, and S. Rathinam, "An approximation algorithm for a task allocation, sequencing and scheduling problem involving a human-robot team," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2146–2153, Apr. 2020.

[11] S. Mayya, D.S. D'antonio, D. Saldaña, and V. Kumar, "Resilient task allocation in heterogeneous multi-robot systems," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1327–1334, Apr. 2021.

[12] Y. Jin, A. A. Minai, and M. M. Polycarpou, "Cooperative real-time search and task allocation in UAV teams," in *Proc. IEEE 42nd Conf. Decis. Control*, Maui, HI, USA, 2003, pp. 7–12.

[13] C. Wei, Z. Ji, and B. Cai, "Particle swarm optimization for cooperative multi-robot task allocation: A multi-objective approach," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2530–2537, Apr. 2020.

[14] D. P. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Comput. Optim. Appl.*, vol. 1, no. 1, pp. 7–66, 1992.

[15] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multirobot task allocation," in *Proc. IEEE/RSI Int. Conf. Intell. Robot. Syst.*, Sendai, Japan, 2004, pp. 698–705.

[16] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 758–768, Oct. 2002.

[17] M. Otte, M. J. Kuhlman, and D. Sofge, "Auctions for multi-robot task allocation in communication limited environments," *Auton. Robots*, vol. 44, no. 3, pp. 547–584, 2020.

[18] M. H. F. B. M. Fauadi, W. Li, and T. Murata, "Combinatorial auction method for decentralized task assignment of multiple-loading capacity AGV based on intelligent agent architecture," in *Proc. IEEE 2nd Int. Conf. Innov. Bio-Inspir. Comput. Appl.*, 2004, pp. 698–705.

[19] K. Zhu et al., "Using a combinatorial auction-based approach for simulation of cooperative rescue operations in disaster relief," *Int. J. Model. Simul. Sci. Comput.*, vol. 9, no. 4, pp. 1–21, 2018.

[20] N. Edalat et al., "Auction-based task allocation with trust management for shared sensor networks," *Secur. Commun. Netw.*, vol. 5, no. 11, pp. 1223–1234, Oct. 2012.

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

[22] H.-L. Choi, A. K. Whitten, and J. P. How, "Decentralized task allocation for heterogeneous teams with cooperation constraints," in *Proc. IEEE Amer. Control Conf.*, Baltimore, MD, USA, 2010, pp. 3057–3062.

[23] A. K. Whitten, H.-L. Choi, L. B. Johnson, and J. P. How, "Decentralized task allocation with coupled constraints in complex missions," in *Proc. IEEE Amer. Control Conf.*, San Francisco, CA, USA, 2011, pp. 1642–1649.

[24] W. Zhao, Q. Meng, and P. W. H. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 902–915, Apr. 2016.

[25] A. Whitbrook, Q. Meng, and P. W. H. Chung, "Reliable, distributed scheduling and rescheduling for time-critical, multiagent systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 732–747, Apr. 2018.

[26] J. Turner, Q. Meng, G. Schaefer, A. Whitbrook, and A. Soltoggio, "Distributed task rescheduling with time constraints for the optimization of total task allocations in a multirobot system," *IEEE Trans. Cybern.*, vol. 48, no. 9, pp. 2583–2597, Sep. 2018.

[27] S. Hunt, Q. Meng, and C. J. Hinde, "An extension of the consensus based bundle algorithm for multi-agent tasks with task based requirements," in *Proc. IEEE 11th Int. Conf. Mach. Learn.*, Baltimore, MD, USA, 2012, pp. 3057–3062.

[28] S. Hunt, Q. Meng, and C. Hinde, "A consensus-based grouping algorithm for multi-agent cooperative task allocation with complex requirements," *Cogn. Comput.*, vol. 6, no. 3, pp. 338–350, Apr. 2014.

[29] W. Zhao, A. Goudar, and A. P. Schoellig, "Finding the right place: Sensor placement for UWB time difference of arrival localization in cluttered indoor environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6075–6082, Jul. 2022.