2024 24th International Conference on Control, Automation and Systems (ICCAS)
Jeju Shinhwa World, Jeju, Korea. Oct 29 - Nov 1, 2024

Distributed Multi-Task Assignment for Multi-Agent Systems

Soya Takamizawa[1*], Toru Namerikawa[2] and Shunsuke Tsuge[3]

[1,2]Graduate School of Science and Technology, Keio University,
Yokohama, Kanagawa, 223-8522, Japan ([1]souyat@nl.sd.keio.ac.jp, [2]namerikawa@sd.keio.ac.jp) * Corresponding author
[3]AI and Flight Control Engineering Section, Kawasaki Heavy Industries, Ltd.,
Kakamigahara, Gihu, 504-8710, Japan (tsuge_shunsuke@global.kawasaki.com)

Abstract: In this research, we solve the task assignment problem of multiple agents to multiple tasks in a multi-agent system. Regarding the problem of one agent versus multiple tasks, we use the concept of "grouping," in which each agent seeks a set of tasks that it can handle at once. On the other hand, for the problem of multiple agents versus one task, we use the concept of the "Timetable" with the goal of synchronizing the start time of each agent's task. By combining the two concepts, it is possible to deal with complex assignment problems, such as each agent executing multiple tasks at the same time, or multiple agents executing one task at the same time. Furthermore, we show that this solution can respond if circumstances changes, such as when the number of agents decreases or the number of tasks increases.

Keywords: Task Assignment, Multi-agent System, Consensus Algorithm, Grouping

1. INTRODUCTION

In recent years, unmanned vehicles have gained popularity and are now used for various tasks like security surveillance operations and delivering packages [1–3]. One key research area is how each agent should be assigned to each target. This is called task assignment problem, and is currently under active investigation [4–7]. One of the solutions which is attracting attention is consensus-based distributed task assignment algorithm [8–15]. It offers advantages like scalability, resilience to faults, and the ability to incorporate designer preferences easily. Among the issues to be considered are complex task assignment problems, such as cases in which multiple agents are required to execute a target task, and single agent can simultaneously execute tasks for multiple targets. [14] proposed an algorithm to find a set of targets that one agent can simultaneously execute. Meanwhile, [15] proposed a distributed task assignment algorithm for situations where there are tasks which require multiple agents simultaneously. It allows multiple agents to execute one task at the same time. As described, although research has been carried out on "one-to-many" assignment problems, "many-to-many" assignment problems have not yet been investigated.

Therefore, the purpose of this research is to address complex assignment problems involving multiple agents and multiple tasks. Each agent initially identifies a set of tasks it can execute concurrently using the concept of grouping [14]. This solves the assignment problem where one agent is assigned to multiple tasks. After that, assignments are made to the grouped set of tasks using Timetable concept proposed in [15]. Timetable is a matrix that each agent possesses individually to specify the time when it begins executing a task. Using Timetable, it becomes possible to execute tasks that require multiple agents and minimize the task start time. Here, in order to support task grouping, two types of Timetables are assigned to each agent. The first one records the start time of individual tasks, similar to the one proposed in [15]. The second one records the execution times of grouped tasks rather than individual tasks. By combining the ideas of grouping and task execution time synchronization, we can address the assignment problem of multiple agents to multiple targets. Furthermore, by leveraging the strength of this method, which specifies the movement of each agent by time, we demonstrate its effectiveness even when circumstances change. Specifically, this method is also applicable when the number of agents decreases or when a new task emerges. The overall flow of the algorithm proposed is illustrated below. The algorithm consists of three types. Algorithm 1 involves the phase where each agent groups tasks. Algorithm 2 is a phase in which each agent creates Timetable decentrally, and Algorithm 3 is a phase in which Timetable is corrected by eliminating conflicts through communication. Initially, Algorithm 1 is executed once, followed by multiple iterations of Algorithms 2 and 3 to achieve convergence.

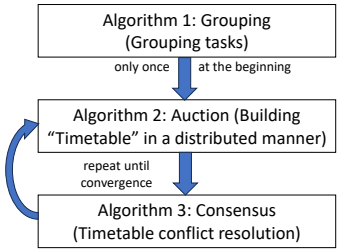

Fig. 1. Conceptual diagram of the entire algorithm.

Section 2 explains our task assignment problem. Section 3 discusses task grouping. In Section 4, assignments are made to the target sets grouped in Section 3. Section 5 demonstrates our method can accommodate changes in the number of agents and tasks. Section 6 presents a simulation, and Section 7 provides a summary.

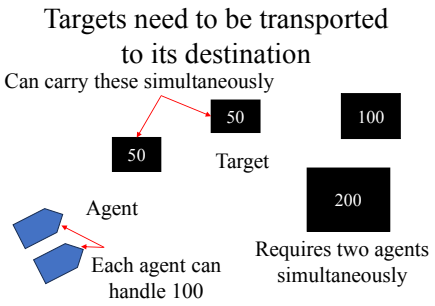

Fig. 2. Problems to consider

2. PROBLEM SETTING

As shown in Fig.2, consider a group of N_a agents corresponding to N_t targets. Small targets can be tackled by one agent at a time, but large targets require multiple agents to tackle them at the same time. It is assumed that all agents are of the same species and there is no difference in their abilities. It is further assumed that each agent can execute tasks a maximum of L_t times, representing a limit on its ability. Additionally, suppose that to excute task k, $R_k \in \mathbb{N}$ agents are required simultaneously. Furthermore, the time required to execute each task is denoted by δ_k. Following [15], our objective is to minimize the average start time of all tasks. Here, in order to show the situation in which each agent can execute multiple tasks simultaneously, in addition to the minimum number of agents R_k required to execute a task, we introduce a new parameter Q_k, which is a real number greater than 0, representing the minimum "capacity" needed to execute a task. We also assume that each agent has the required ability W_i to execute its task. Each agent can simultaneously execute tasks within a range where the sum of Q_k does not exceed W_i. However, since the agents are of the same type, W_i is the same for all agents. Therefore, $W_i = W_t$. Regarding Q_k, W_t and R_k, the relationship $W_t \times R_k \geq Q_k$ must be satisfied.

3. ALGORITHM1 : GROUPING[14]

Firstly, with reference to [14], we present the method to determine the set of tasks that each agent i can handle simultaneously. The fundamental concept revolves around the notion that if tasks can be executed collectively, they can also be executed individually. For instance, if three tasks A, B, and C can be handled simultaneously, it follows that pairs such as A and B, A and C, and B and C can also be managed concurrently, as well as each task individually. In the actual algorithm, the reverse calculation is conducted. For the specific algorithm, please refer to [14].

Here, assign a numerical identifier to each subset of $\mathcal{F}^i$ and arrange them in order. This step is essential for referencing the elements of each subset of $\mathcal{F}^i$ in the subsequent algorithm.

4. ALGORITHM2,3 : CONSENSUS ALGORITHM

4.1 Concept of Timetable[15]

In [15], the concept of Timetable is introduced to allow each agent to execute tasks simultaneously. When multiple agents are required to execute a task simultaneously, the agent reaching the task location first must wait to commence the task until the necessary number of agents is present. Consequently, the actual start time of a task corresponds to the moment when the last agent assigned to that task arrives at the designated location. Therefore, in order to determine the actual start time of each task, each agent needs to know the estimated start times of all other agents' tasks.

Timetable $\mathbf{T}^i = [T^i_{jk}] \in \mathbb{R}^{N_a \times N_t}$ is a matrix where each agent i records the estimated task start time of all agents, including itself. T^i_{jk} indicates the value of the j-th row and k-th column of $\mathbf{T}^i$. $T^i_{jk} > 0$ indicates that agent i estimates that agent j will start executing task k at time T^i_{jk}. Furthermore, $T^i_{jk} = 0$ indicates that agent i estimates that agent j will not execute task k. When the Timetables of all agents are equal, each agent can execute tasks according to its own Timetable. Let $\mathbf{a}^i_k$ be the set of agents corresponding to task k estimated by agent i, τ^i_k be the actual start time of task k estimated by agent i, and $\mathbf{b}^i_j$ be the set of tasks to be executed by agent j estimated by agent i, each case can be expressed using T^i_{jk} as follows.

$$\mathbf{a}^i_k = \{j \in \mathcal{A} | T^i_{jk} > 0\} \tag{1}$$

$$\tau^i_k = \max_{j=1}^{N_a} T^i_{jk} \tag{2}$$

$$\mathbf{b}^i_j = \{k \in \mathcal{T} | T^i_{jk} > 0\} \tag{3}$$

Furthermore, the sequence $\mathbf{p}^i_j$ of the order in which agent j performs tasks, as estimated by agent i, can be found by arranging T^i_{jk} corresponding to each element of $\mathbf{b}^i_j$ in descending order.

4.2 New Timetable

In this paper, for each of the five variables $\mathbf{T}^i$, τ^i_k, $\mathbf{a}^i_k$, $\mathbf{b}^i_k$, and $\mathbf{p}^i_k$ explained in Sec.4.1 , we will distinguish between two types: when each task is considered individually and when they are considered as a group. The reason for this distinction is that, as explained later, the cost function formula, etc., is computed when each task is considered individually. Add subscript "g" if you are referring to a group, and add subscript "o" if you are referring to an individual. An example of Timetable $\mathbf{T}g^i \in \mathbb{R}^{N_a \times |\mathcal{F}^i|}$ when tasks are considered as groups is illustrated.

Suppose that agent 2 groups the four tasks according to Grouping above and obtains a set $\mathcal{F}^2 = \{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}\}$. At this time, assign a number corresponding to each subset of $\mathcal{F}^2$. Specifically, let $\{1\}$ be 1, $\{2\}$ be 2, $\{3\}$ be 3, $\{4\}$ be 4, and $\{1, 2\}$ be 5. In this case, the Timetable $\mathbf{T}g^2$ corresponding to grouping is a matrix with N_a rows and $|\mathcal{F}^2|$ columns. Suppose that $N_a = 3$. In this case, suppose that agent $i = 2$ has

$\mathbf{T}g^2 \in \mathbb{R}^{3\times 4}$ as follows.

$$\mathcal{F}^2 = \{\{1\},\{2\},\{3\},\{4\},\{1,2\}\} \tag{4}$$

$$\mathbf{T}g^2 = \begin{bmatrix} 0 & 0 & 8 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 \end{bmatrix} \tag{5}$$

Each line of $\mathbf{T}g^2$ corresponds to an agent number. However, each column does not correspond to a task itself, but to a number assigned to each subset of $\mathcal{F}^2$. In other words, each column corresponds to a set of tasks after grouping.

For example, if the element in the 1-st row and 3-rd column of $\mathbf{T}^2$ is 8, agent 2 estimates that agent 1 will start executing the 3-rd element of the subset of $\mathcal{F}^2$ at time 8 seconds. This corresponds to task 3. On the other hand, if the element in the 1-st row and 5-th column of $\mathbf{T}^2$ is 4, agent 2 estimates that agent 1 will start executing the 5-th element of the subset of $\mathcal{F}^2$ at time 4 seconds. This corresponds to task 1, 2. In other words, agent 2 estimates that agent 1 will start executing tasks 1 and 2 at time 4 seconds.

Even when considered in grouping, the set of agents corresponding to task k that agent i estimates at this point $\mathbf{a}g_k^i$, the actual start time of task k τg_k^i, the set of tasks that agent j executes $\mathbf{b}g_j^i$, and the sequence of the order in which agent j executes tasks $\mathbf{p}g_j^i$ can be calculated in the same way as in Sec.4.1. That is, it is expressed as below.

$$\mathbf{a}g_k^i = \{j \in \mathcal{A} | Tg_{jk}^i > 0\} \tag{6}$$

$$\tau g_k^i = \max_{j=1}^{N_a} Tg_{jk}^i \tag{7}$$

$$\mathbf{b}g_j^i = \{k \in \mathcal{T} | Tg_{jk}^i > 0\} \tag{8}$$

As before, $\mathbf{p}g_j^i$ can be found by arranging Tg_{jk}^i corresponding to each element of $\mathbf{b}g_j^i$ in descending order. Once the Timetable $\mathbf{T}g^i$ is found, the corresponding τg_k^i, $\mathbf{a}g_k^i$, $\mathbf{b}g_j^i$, and $\mathbf{p}g_k^i$ can also be found.

4.2.1 Relationship between $\mathbf{T}g^i$ and $\mathbf{T}o^i$

We will explain the relationship between the two Timetables $\mathbf{T}g^i \in \mathbb{R}^{N_a \times |\mathcal{F}^i|}$ and $\mathbf{T}o^i \in \mathbb{R}^{N_a \times N_t}$. $\mathbf{T}o^i$ records the start time of each task without grouping them. Therefore, $\mathbf{T}o^i$ is expressed as if the elements of the grouped parts of each value of $\mathbf{T}g^i$ were also executed individually. In other words, $\mathbf{T}o^i$ corresponding to Eq.(5) is expressed as follows. Task group numbers 1, 2, and 3 indicate one task, so the corresponding elements remain unchanged, but task group number 4 is a grouping of tasks 1 and 2. Therefore, tasks 1 and 2 of the corresponding agent 1 are updated with the value of task group number 4.

$$\mathcal{F}^2 = \{\{1\},\{2\},\{3\},\{4\},\{1,2\}\} \tag{9}$$

$$\mathbf{T}g^2 = \begin{bmatrix} 0 & 0 & 8 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 \end{bmatrix} \quad \mathbf{T}o^2 = \begin{bmatrix} 4 & 4 & 8 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 7 \end{bmatrix} \tag{10}$$

At this time, the set of agents corresponding to task k that agent i estimates at this point $\mathbf{a}o_k^i$, the actual start time of task k τo_k^i, the set of tasks that agent j executes $\mathbf{b}o_j^i$, and the sequence of the order in which agent j executes tasks $\mathbf{p}o_j^i$ can be calculated in the same way as in Sec.4.1.

$$\mathbf{a}o_k^i = \{j \in \mathcal{A} | To_{jk}^i > 0\} \tag{11}$$

$$\tau o_k^i = \max_{j=1}^{N_a} To_{jk}^i \tag{12}$$

$$\mathbf{b}o_j^i = \{k \in \mathcal{T} | To_{jk}^i > 0\} \tag{13}$$

As before, $\mathbf{p}o_j^i$ can be found by arranging To_{jk}^i corresponding to each element of $\mathbf{b}o_j^i$ in descending order. $\mathbf{T}o^i$ is uniquely determined by $\mathbf{T}g^i$, and the corresponding τg_k^i, $\mathbf{a}g_k^i$, $\mathbf{b}g_j^i$, and $\mathbf{p}g_k^i$ can also be found.

4.3 Problems to be solved

The problem that each agent i needs to solve decentrally is expressed as follows by advancing the start time of all tasks. Cost function will be described in Eq.(20).

$$\min_{\mathbf{T}g^i} S_i(\mathbf{T}o^i) \tag{14}$$

$s.t.$

$$\mathbf{T}g^i = \mathbf{T}g^j, \forall (i,j) \in \mathcal{A}^2, i \neq j \tag{15}$$

$$\sum_{j=1}^{N_a} h(Tg_{jk}^i > 0) = R_k, \forall (i,k) \in \mathcal{A} \times \mathcal{T} \tag{16}$$

$$\sum_{j=1}^{N_a} W_t h(Tg_{jk}^i > 0) \geq Q_k, \forall (i,k) \in \mathcal{A} \times \mathcal{T} \tag{17}$$

$$\sum_{k=1}^{N_t} h(Tg_{jk}^i > 0) \leq L_t, \forall (i,j) \in \mathcal{A}^2 \tag{18}$$

Eq.(14) aims to minimize the average start time of all tasks. Eq.(15) shows that Timetables of all agents are equal. Eq.(16) and Eq.(17) are constraints on the number and capacity of agents, respectively. Eq.(18) is a constraint on the upper limit of the number of tasks assigned to agent i. $h(\cdot)$ is a binary variable.

We will explain two Timetables construction algorithms and a consensus algorithm according to [15]. In reality, equation Eq.(14) cannot be solved directly.

4.4 Timetable construction algorithm

From $\mathbf{a}g_k^i$ and $\mathbf{T}g^i$, the task start time $\tau^i g_k$ can also be found as follows:

$$\tau g_k^i = \max_{j \in \mathbf{a}g_{ik}} \mathrm{T}g_k^i \tag{19}$$

If i thinks that another agent j will also execute task k, it indicates that the maximum value is the start time.

Next, we explain the cost function. When Timetable $\mathbf{T}o_i$ is given, and the cost function estimated by i is $S_i(\mathbf{T}o_i)$, the cost function for her that satisfies the task requirements and also has an earlier start time can be expressed as follows. U is a sufficiently large number compared to the sum of τo_{ik}. That is, let $U > \sum_{k=1}^{N_t} \tau o_{ik}, \forall i \in \mathcal{A}$.

$$S_i(\mathbf{T}o^i) = \sum_{k=1}^{N_t} \left\{ \left| R_k - \left| \mathbf{a}o_k^i \right| \right| \cdot U + \tau o_k^i \right\} \tag{20}$$

Eq.(20) is a function of the Timetable $\mathbf{T}o^i$ when considering each task individually, but since $\mathbf{T}o^i$ is uniquely determined from $\mathbf{T}g^i$, the cost function is also a function of $\mathbf{T}g^i$. Timetable $\mathbf{T}^i$ is defined as follows. p^i_m indicates the mth element of $\mathbf{p}^i$.

$$\mathrm{T}g^i_{ip^i_m} = \begin{cases} \frac{d_a\{i,p^i_m\}}{v}, & \text{if } m=1 \\ \tau g^i_{p^i_{(m-1)}} + \delta_{p^i_{(m-1)}} + \frac{d_t\{p^i_{(m-1)},p^i_m\}}{v}, & \text{otherwise} \end{cases} \tag{21}$$

Here, $d_a\{i,p^i_m\}$ indicates the distance between the initial position of agent i and task p^i_m. Also, $d_t\{p^i_{(m-1)},p^i_m\}$ indicates the distance between the execution point of task $p^i_{(m-1)}$ and task p^i_m. Also, v indicates the speed of agent i.

Here, the reduction in the cost function when task k is added before the lth task in the execution sequence $\mathbf{p^i}$ is defined as follows. $\mathbf{T}^i \oplus_l k$ denotes agent i's Timetable $\mathbf{T}^i$ when he inserted task k after the lth element.

$$c_{ik}\left(\mathbf{T}g^i\right) = \begin{cases} -U, & \text{if } Tg^i_{ik} > 0 \text{ or } r^i > \gamma \\ \max_{l=1}^{|\mathbf{p}^i|+1} \left\{S_i(\mathbf{T}g^i) - S_i(\mathbf{T}g^i \oplus_l k)\right\}, & \text{otherwise} \end{cases} \tag{22}$$

If the number of removed tasks exceeds the threshold, or if i already thinks that task k will be executed by itself, the cost function increases.

The overall flow is that for each $\mathcal{G}$ in which i is newly executable, based on Eq.(22), calculate the desired task insertion position l and the reduction width c_{ik} of the cost function at that time. Then, for k that gives the maximum c_{ik} , Timetable $\mathbf{T}_i$ etc. are updated using k and the corresponding l.

Algorithm 1 Agent i build Timetable decentrally

while $\sum_{k=1}^{|\mathcal{F}^i|} h(Tg^i_{ik} > 0) \leq L_t$ **do**
  for all $k \in \mathbb{N}, 1 \leq k \leq |\mathcal{F}^i|$ **do**
    if $l \notin \mathbf{p}g^i_i, \forall l \in \mathcal{F}^i_k$ **then**
      compute c_{ik}
    end if
  end for
  $g = \max c_{ik}$
  if $g \geq 0$ **then**
    $q = \mathrm{argmax}\, c_{ik}$
    $n = \mathrm{argmax}_{l \leq |\mathbf{p}_i|+1}\{S_i(\mathbf{T}g_i) - S_i(\mathbf{T}g_i \oplus_l q)\}$
    Find $\mathbf{a}^i_q$ according to $\mathbf{T}g_i \oplus_l q$
    if $i \in \mathbf{a}^i_q$ **then**
      update Timetable,$\mathbf{T}g^i \leftarrow \mathbf{T}g^i \oplus_n q$
    else
      break
    end if
  else
    break
  end if
end while

4.5 Consensus Algorithm

Since the Timetable is calculated in a distributed manner, conflicts may occur, such as when the requirements for the number of tasks do not meet the conditions. Therefore, we aim to eliminate conflicts and make Timetables consistent among all agents through a consensus algorithm through communication. First, in the first part, agent i communicates with the Timetable and timestamps$^i = [s^i_j] \in \mathbb{R}^{N_a}$ according to the communication graph. Information is received directly from neighboring agents and indirectly from other agents.

Algorithm 2 Consensus Phase

Send $\mathbf{T}g^i$ and $\mathbf{s}^i$ to agent j if $G_{ij} = 1$
Receive $\mathbf{T}g^j$ and $\mathbf{s}^j$ from agent j if $G_{ij} = 1$
$m = \mathrm{argmax}_{u=1}^{Na} \mathrm{max} s^m_u, G_{iu} \neq 1 \wedge G_{im} = 1$
for $j = 1$ to N_a **do**
  if $G_{ij} = 1$ **then**
    $Tg^i_{jk} = Tg^j_{jk}, \forall k \in \mathcal{T}$
  else
    $Tg^i_{jk} = Tg^m_{jk}, \forall k \in \mathcal{T}$
  end if
end for
for $k = 1$ to Nt **do**
  while $\sum_{j=1}^{Na} h(Tg^i_{jk} > 0) > R_k$ **do**
    $q = \mathrm{argmax}_{j=1}^{N_a} Tg^i_{jk}$
    $Tg^j_{qk} = 0$
    if $q = i$ **then**
      $r^i = r^i + 1$
    end if
  end while
end for
compute $\mathbf{T}g^i$

In the second part, for those exceeding the task number requirement R_k, the Timetables are deleted starting from the one with the maximum $\mathbf{T}g$. When its own task is deleted, the number of deleted tasks r^i is updated. Regarding the number of tasks removed, this is done to prevent the algorithm from not converging and falling into an infinite loop.

If we can calculate the Timetable $\mathbf{T}g^i$ when tasks are considered as a group, we can also derive the Timetable $\mathbf{T}o^i$ etc. when tasks are considered individually.

5. APPLICATION WHEN THE SITUATION CHANGES

Let us consider applying this algorithm to cases where the situation changes. The following two cases can be considered when the situation changes.

- When a new task is added
- When the agent runs out (breaks down)

Here, we make the following assumptions. This assumption indicates that all agents can instantly know that a new task has been found or that an agent has failed.

Assumption 1: The graphs of all normal agents are connected. In other words, the agent's communication range is infinite.

5.1 When a new task is added

First, consider the case where a new task is added. In this paper, we propose that when a new task is found, re-

allocation should be performed so that the cost function becomes smaller. In other words, we propose to allocate tasks that have not yet been executed at the time a new task is found. Here, let $time[s]$ be the time when a new task is found. As a specific algorithm, when tasks are found, the tasks are first grouped using Grouping. Next, each agent derives $\mathbf{T}$ from the initial time using Algo.3. After that, Algo.2 is used to derive $\mathbf{T}$ without duplication from the initial time. Repeat Algo.3 and Algo.2 until convergence.

Algorithm 3 When new task is discovered

1: **while** $\sum_{k=1}^{|\mathcal{F}^i|} h(Tg_{ik}^i > 0) \leq L_t$ **do**
2: **for all** $k \in \mathbb{N}, 1 \leq k \leq |\mathcal{F}^i|$ **do**
3: **if** $time < \tau g_{ik}^i$ **then**
4: compute c_{ik}
5: **end if**
6: **end for**
7: Compute Algo.1's 7line-19line
8: **end while**

τg^i indicates the time when the task started. Therefore, the third line of Algo.3 means that the cost function is calculated for tasks that have not yet started execution.

5.2 When the agent runs out (breaks down)

When an agent fails, consider reassigning tasks that were assigned to the failed agent but have not yet been executed to the remaining agents. In this case, as in the case where a new task is found in the previous section, $\mathbf{T}$ from the initial time is derived by repeating the auction algorithm shown below and the consensus algorithm in Algo.2. Let time be the $time[s]$ when the failure occurred, and let j be the agent where the failure occurred. In this case, the consensus algorithm is expressed as follows.

Algorithm 4 When one agent fails

1: **while** $\sum_{k=1}^{|\mathcal{F}^i|} h(Tg_{ik}^i > 0) \leq L_t$ **do**
2: **for all** $k \in \mathbb{N}, 1 \leq k \leq |\mathcal{F}^i|$ **do**
3: **if** $time < \tau g_{jk}^i + \delta_{pg_j^i}$ **then**
4: compute c_{ik}
5: **end if**
6: **end for**
7: Compute Algo.1's 7line-19line
8: **end while**

$\tau g_{ik}^i + \delta_{p_k^i}$ indicates the time when the task finished executing. Therefore, the third line of Algo.4 means that the cost function is calculated for tasks that have not yet been executed.

6. SIMULATION

We perform a simulation of the proposed method and compare it with [15], which allocates multiple agents to one task. Consider a $100[m]$ square plane in which multiple agents transport cargo scattered on the plane to the destination. The number of task executions is defined as the number of times the destination is reached. Therefore, L_t is the limit on the number of times the agent can go to the destination.

$N_a = 4$,$N_t = 9$, and we set $L_t = 3$,$W_t = 1$. The speed of the agent is set to $v = 1[m/s]$ and $U = 10000$. Q_k is determined as follows, and R_k is the minimum integer that satisfies $W_t \times R_k \geq Q_k$. At this time, when tasks are grouped, $\mathcal{F}^i$ is obtained as follows.

$$Q_k = [0.3 \quad 0.3 \quad 0.3 \quad 0.5 \quad 0.5 \quad 1 \quad 1 \quad 2 \quad 2] \tag{23}$$

$$\begin{aligned}\mathcal{F}^i = \Big\{&\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\\ &\{1,2\},\{1,3\},\{1,4\},\{1,5\},\{2,3\},\{2,4\},\\ &\{2,5\},\{3,4\},\{3,5\},\{4,5\},\{1,2,3\}\Big\}\end{aligned} \tag{24}$$

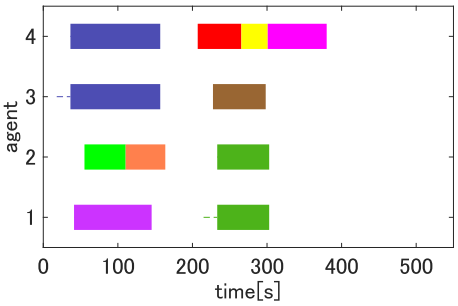

Fig. 3. Proposed method

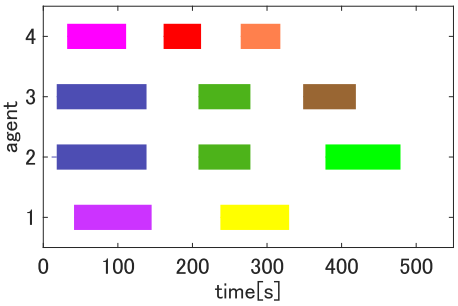

Fig. 4. [15]'s method

The results of Algo.1 and Algo.2 are shown in the figure. A thin dashed line indicates that an agent has arrived early and is waiting for other agents. In the proposed method, one agent executes multiple tasks simultaneously, so the task completion time is earlier than in the conventional method. In Fig.3, agent 1 appears to be carrying four packages, but since three of these packages are being carried at the same time, the number of times the task is executed is two.

Next, we will show how to respond to the changes in the situation shown in Sec.5. After making the initial assignment time shown in Fig.3, suppose an event occurs at 100 seconds while the system is actually running. Regarding agent failure, suppose agent 2 has failed. Regarding new tasks, suppose a task that requires only one agent

has been discovered. Each figure shows the allocation results corresponding to each event.

In Fig.5, we can see that other agents are performing the task that was being handled by the failed agent 2. In Fig.6, we can see that the assignment has been reassigned so that the newly discovered task, represented in emerald green, can be handled. The black lines in each figure indicate the time when the event occurred.

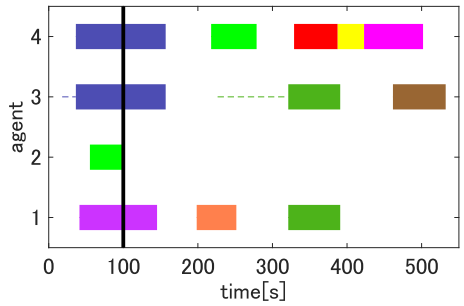

Fig. 5. Reassignment when agent 2 is broken

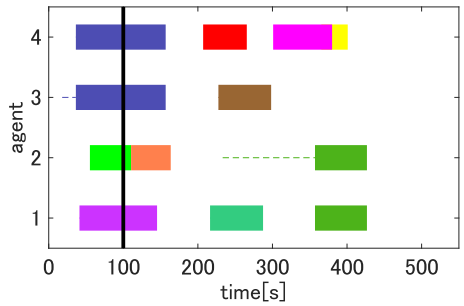

Fig. 6. Reassignment when new task is discovered

7. CONCLUSION

In this study, we investigated the problem of task assignment between multiple agents and multiple targets. Specifically, we combined the concept of grouping, in which each agent seeks a set of targets that it can simultaneously respond to, and the concept of synchronizing the task start times of multiple agents. This makes it possible to handle situations in which one agent is responding to multiple targets at the same time, or situations in which multiple agents are responding to one target at the same time. We also applied the proposed method to cases where the environment changes, making it possible to respond to increases or decreases in the number of agents or targets.

REFERENCES

[1] M. Erdelj, E. Natalizio, K. R. Chowdhury and I. F. Akyildiz, "Help from the Sky: Leveraging UAVs for Disaster Management," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 24-32, 2017.

[2] D. Patil et al., "A Survey On Autonomous Military Service Robot," in *Proc. of 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pp. 1-7, 2020.

[3] K. Kuru, D. Ansell, W. Khan and H. Yetgin, "Analysis and Optimization of Unmanned Aerial Vehicle Swarms in Logistics: An Intelligent Delivery Platform," *IEEE Access*, vol. 7, pp. 15804-15831, 2019.

[4] H. Shakhatreh et al., "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," *IEEE Access*, vol. 7, pp. 48572-48634, 2019.

[5] 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.

[6] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li and L. Wang, "Joint Optimization of Multi-UAV Target Assignment and Path Planning Based on Multi-Agent Reinforcement Learning," *IEEE Access*, vol. 7, pp. 146264-146272, 2019.

[7] M. Machida and M. Ichien, "Consensus-Based Artificial Potential Field Approach for Swarm," in *Proc. of 60th IEEE Conference on Decision and Control (CDC)*, pp. 6671-6677, 2021.

[8] H. -L. Choi, L. Brunet and J. P. How, "Consensus-Based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912-926 2009.

[9] L. Luo, N. Chakraborty and K. Sycara, "Distributed As for Multirobot Task Assignment With Task Deadline Constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876-888, 2015.

[10] X. Duan, H. Liu, H. Tang, Q. Cai, F. Zhang and X. Han, "A Novel Hybrid Auction Algorithm for Multi-UAVs Dynamic Task Assignment," *IEEE Access*, vol. 8, pp. 86207-86222, 2020 .

[11] 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.

[12] S. Raja, G. Habibi and J. P. How, "Communication-Aware Consensus-Based Decentralized Task Allocation in Communication Constrained Environments," *IEEE Access*, vol. 10, pp. 19753-19767, 2022.

[13] W. Wu, J. Xu and Y. Sun, "Integrate Assignment of Multiple Heterogeneous Unmanned Aerial Vehicles Performing Dynamic Disaster Inspection and Validation Task With Dubins Path," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 4, pp. 4018-4032, 2023.

[14] 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.

[15] S. Wang, Y. Liu, Y. Qiu and J. Zhou, "Consensus-Based Decentralized Task Allocation for Multi-Agent Systems and Simultaneous Multi-Agent Tasks," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12593-12600 2022.