J Intell Robot Syst (2017) 87:97–123
DOI 10.1007/s10846-017-0493-x

Market-Based Task Assignment for Cooperative Timing Missions in Dynamic Environments

Gyeongtaek Oh · Youdan Kim · Jaemyung Ahn · Han-Lim Choi

Received: 29 January 2016 / Accepted: 18 January 2017 / Published online: 30 January 2017

Abstract New market-based decentralized algorithms are proposed for the task assignment of multiple unmanned aerial vehicles in dynamic environments with a limited communication range. In particular, a cooperative timing mission that cannot be performed by a single vehicle is considered. The baseline algorithms for a connected network are extended to deal with time-varying network topology including isolated subnetworks due to a limited communication range. The mathematical convergence and scalability analyses show that the proposed algorithms have a polynomial time complexity, and numerical simulation results support the scalability of the proposed algorithm in terms of the runtime and communication burden. The performance of the proposed algorithms is demonstrated via Monte Carlo simulations for the scenario of the suppression of enemy air defenses.

G. Oh
Seoul National University, Seoul, 08826, Republic of Korea
e-mail: laimo@snu.ac.kr

Y. Kim (✉)
The Institute of Advanced Aerospace Technology,
Seoul National University, Seoul, 08826, Republic of Korea
e-mail: ydkim@snu.ac.kr

J. Ahn · H.-L. Choi
Korea Advanced Institute of Science and Technology,
Daejeon, 34141, Republic of Korea
e-mail: jaemyung.ahn@kaist.ac.kr

H.-L. Choi
e-mail: hanlimc@kaist.ac.kr

Keywords Multi UAV · Market-based task allocation · Cooperative timing mission · Dynamic environment

1 Introduction and Related Work

With increasing demand for unmanned aerial vehicles (UAVs) in military and civilian areas, coordination of multiple UAVs is expected to play a key role in complex missions. However, there exist several technical issues in employing multiple UAVs. As the number of agents and tasks increase, a greater burden is imposed on ground operators, which may cause safety issues. Due to the limited communication range of UAVs, the network topology varies in real time and sometimes may be disconnected. Additionally, in a disaster area or a battlefield, the mission environment often changes dynamically. In this study, the autonomous and decentralized task allocation (TA) problem in dynamic environments is considered. In particular, a type of task simultaneously requiring multiple UAVs is considered.

Many studies have solved a TA problem using centralized [1, 6, 16, 23, 35, 38] and decentralized [2, 7–9, 22, 24, 36] approaches; however, most considered tasks requiring a single agent. Tightly coupled tasks such as simultaneous attack and box pushing, on the other hand, require multiple agents for maximizing the effectiveness of the multi-agent system to perform the given mission. According to the

taxonomy presented by Gerkey and Matarić [19], the former problem is called the single-task robots (ST) – single-robot tasks (SR) problem, and the latter problem is called the ST - multi-robot tasks (MR) problem.

The ST-MR problem is referred to as a *coalition* formation problem where the coalition means a subgroup of agents to conduct a common task [33]. Note that the primary focus of this paper is the overlapping coalition, in that agents are allowed to be members of multiple coalitions. Shehory and Kraus [37] presented the iterative greedy algorithm for the coalition formation that operates in exponential time. Experimental demonstrations of the distributed auction-based approach for the box pushing problem [18, 44], cooperative load transportation [27], and disaster management [24] were respectively presented. The coalition formation for the simultaneous attack problem was treated using a centralized [42] and distributed scheme [25, 41]. The centralized scheme solved the combinatorial optimization problem for the coalition formation using the particle swarm optimization (PSO) technique, and the distributed scheme utilized an auction and an integer programming. Modification of the Shehory and Kraus' algorithm and associated complexity analysis were presented for a non-overlapping coalition case [34]. A consensus-based bundle algorithm (CBBA) [9] was extended to coupled-constraint CBBA [46] to consider tightly coupled tasks. The bio-inspired coalition formation approach was proposed to apply the suppression of enemy air defenses (SEAD) mission [21]. Das et al. proposed market-based coalition formation that allocates multiple tasks both in a centralized [11] and distributed manner [12]. However, the aforementioned coalition formation algorithms do not consider a dynamic network topology and limited communication range. When a distributed algorithm runs in a dynamic network, TA results depend on the communication range.

Several TA studies addressed the problem of limited communication range. Beard and McLain [5] proposed a centralized cooperative path planning method considering distance constraints between UAVs, and Sujit and Beard [40] presented a distributed auction algorithm over a limited communication range. CBBA was extended to ensure network connectivity because a limited communication range may result in a disconnected network [30]. Another idea to overcome the loss of network connectivity is to make the idle agent return to the base[31]. Whereas the aforementioned research considered the ST-SR problem, studies on the ST-MR problem over a limited communication range have also been performed. Weerdt et al. [45] proposed a variant of contract net protocol (CNP) [39] for the coalition formation over a limited communication range using the distributed sequential auction, but the qualification of a coalition leader (auctioneer) was not considered. On the other hand, the decentralized coalition formation algorithm for UAVs to track and destroy moving targets [17, 43] are proposed with an extensive numerical analysis of the effect of communication ranges, delays, and the problem size; however, neighboring UAVs must share their position and path information continuously to predict that their subnetwork is invariant during the coalition formation process.

In this study, two market-based decentralized coalition formation algorithms in dynamic environments with limited communication ranges are proposed; one is a project manager-oriented coalition formation algorithm (PCFA) and the other is a task-oriented coalition formation algorithm (TCFA). Because the theoretical basis of the market-based approach [14] is established utilizing CNP, the proposed algorithm can be classified as a variant of CNP or a distributed auction. The agents in a virtual market negotiate with *neighborhood*, defined as agents within their communication range, using local information to maximize the group utility. The virtual market follows the designed market mechanism, which consists of four intuitive phases, working in parallel over all of the agents. Mathematical and numerical analyses in the connected network are carried out to show the convergence, polynomial time complexity, and bounded communication burden of the proposed methods. The proposed algorithms are then extended to deal with the problem in dynamic environment, where the network connectivity between agents depends on their distance. The performance of the proposed algorithms in a dynamic environment is demonstrated by numerical simulations considering multiple UAV SEAD missions with pop-up tasks.

The main contribution of this work can be summarized as follows. First, two new decentralized TA algorithms are proposed by extending the decentralized TA algorithm in a static environment [29] to a dynamic environment with a limited communication range. In the proposed algorithms, a leader of a coalition is elected by other agents, and each agent's position

and plan do not have to be continuously synchronized through agents. Second, analyses on convergence and scalability are performed, which are also supported by numerical results. Finally, the performance of the proposed algorithms are demonstrated through Monte Carlo simulations and compared with the state-of-the-art algorithm, which is modified by this study.

The paper is organized as follows. Two TA algorithms are proposed in Section 2 with a description of the TA problem. In Section 3, properties of the proposed algorithms are analyzed. In Section 4, the proposed algorithms are extended to dynamic environments. Numerical results are shown in Section 5. Finally, conclusions are given in Section 6.

2 TA Algorithm for Cooperative Timing Missions over Connected Networks

2.1 Problem Statement

Let us consider a TA problem with N agents and M tasks. The solution for the TA problem should provide the *best* sequences of tasks for each agent to maximize the objective function. In particular, some task locations should be simultaneously visited by multiple agents as requirements [28]. The number of required agents for each task is predefined based on its degree of risk. The problem considered can be formulated in terms of the integer programming problem as follows:

$$\underset{\mathbf{p}^{(i)},\forall i\in\mathcal{I}}{\text{Maximize}} J = \sum_{k=1}^{M} s_k(t_k(\mathbf{P})) \tag{1}$$

$$\text{subject to } n(\mathbf{a}_k(\mathbf{P})) = Z_k,\ \forall k \in \mathcal{K} \tag{2}$$

$$n\left(\mathbf{p}^{(i)}\right) \le L_i,\ \forall i \in \mathcal{I} \tag{3}$$

$$G(\mathcal{K}, \mathcal{E}(\mathbf{P})) = \text{DAG} \tag{4}$$

where $\mathcal{I} \triangleq \{1, ..., N\}$ and $\mathcal{K} \triangleq \{1, ..., M\}$ are the index sets of the agents and tasks, respectively, $\mathbf{p}^{(i)}$ is a path vector representing the sequence of tasks allocated to the agent i, and $\mathbf{P}$ is the path list of all of the agents. The function s_k represents the score when the task k is performed at time t_k where t_k is the termination time of task k. Eq. 2 defines a constraint on the size of the coalition for task k where $\mathbf{a}_k$ is a set of indices of agents assigned to task k, $n(\cdot)$ denotes the cardinality of a set, and Z_k is the number of agents required for the task k. Eq. 3 restricts the number of allowable tasks for each agent, where L_i is the maximum number of allowable tasks for the agent i.

Additionally, Eq. 4 is introduced to disregard the TA results, of which involved agents fail to simultaneously arrive at their allocated task locations. This process can be conducted by filtering out the TA results that generate the directed cycle in the dependency graph $G(\mathcal{K}, \mathcal{E}(\mathbf{P}))$, which represents the priority between the allocated tasks [4]. Because the simultaneous arrival fails when the dependency graph contains a directed cycle, designating the type of dependency graph as a directed acyclic graph (DAG), which is a directed graph with no directed cycles [20], realizes the filtering. More detailed explanations on the DAG constraint and the directed edge set $\mathcal{E}(\mathbf{P})$ are provided in Appendix A.

The objective function J indicates the total mission score. To consider completion time as well as the priority between tasks, the score function s_k can be defined as follows.

$$s_k(t_k(\mathbf{P})) = w_k e^{-\lambda_k(t_k - t_k^0)} \tag{5}$$

where w_k is the inherent worth of the task k, $\lambda_k > 0$ is the time-discounting factor for the task k, and t_k^0 is the time when task k is generated. Because some tasks should be visited at the same time, t_k is defined as the arrival time of the latest agent among the coalition for task k, which is expressed as

$$t_k(\mathbf{P}) = \underset{i\in\mathbf{a}_k}{max}\ \ (t_{ETA}(i,k) + t_w(i,k)) \tag{6}$$

where $t_w(i,k)$ denotes the required working time of agent i to perform task k and $t_{ETA}(i,k)$ denotes the estimated time of arrival (ETA) of the agent i to the task k, which can be expressed as

$$t_{ETA}(i,k) = \begin{cases} d_{ik}/v_i, & \text{if } \mathbf{p}^{(i)}(1) = k \\ t_l + d_{lk}/v_i, & \text{otherwise} \end{cases} \tag{7}$$

where v_i is the average speed of agent i. If $\mathbf{p}^{(i)}(1) = k$, that is, task k is the first task of agent i, then d_{ik} is the distance between the current location of agent i and task k. Otherwise, d_{lk} is the distance between task l and task k, where l is the index of the task conducted by agent i just before task k, which can be expressed as

$$l = \mathbf{p}^{(i)}(b_k^{(i)} - 1) \tag{8}$$

where $\mathbf{p}^{(i)}(b)$ is the b-th element of $\mathbf{p}^{(i)}$, and index $b_k^{(i)}$ satisfies the following relationship:

$$\mathbf{p}^{(i)}(b_k^{(i)}) = k \tag{9}$$

The TA problem defined in Eqs. 1∼4 is a coalition formation problem where coalitions may overlap, and this problem can be categorized as a set covering problem (SCP) [37]. The original SCP is known as NP-hard [10]. Because the original SCP does not consider the order of task allocation associated with Eq. 4, our problem is at least as complex as the SCP. When the size of the problem becomes larger, the centralized algorithm requires a significant amount of time to determine the exact optimal solution, which is not appropriate for the dynamic environment. In this section, decentralized market mechanisms are presented to solve the complex TA problem in a suboptimal approach with less computational/communicational burden.

2.2 Assumption

Through this section, the following assumptions are considered.

- The network topology between agents is a connected graph that there exists a path between every pair of vertices. Some pairs of vertices may not be directly connected due to the limited communication range, but there are no unreachable vertices.
- The network topology does not change during the process of TA.
- Agents communicate with each other in a synchronous manner, i.e., each agent communicates according to the scheduled time table.

The first and second assumptions can be accepted when the communication range is sufficiently long and the required time for TA is small. In the later section on extended algorithms for dynamic environment, these two assumptions will be omitted. The last assumption may not be appropriate for real-world application because asynchronous communication is more efficient. However, for the purpose of the analysis of the proposed algorithms, synchronous communication is considered in this study.

2.3 Project Manager-oriented Coalition Formation Algorithm: PCFA

2.3.1 Preliminaries

Let us consider a virtual market consisting of N agents and M tasks. Because the task $k \in \mathcal{K}$ requires Z_k agents to be performed, agents should build several temporary coalitions, where the team members may be overlapped. The goal of this study is to design a rule for each agents to allocate the given tasks by themselves based on the communication between agents.

Suppose that the tasks and the agents are interpreted as the projects and the contractors [39]. In PCFA, agents once make consensus on both a project manager (PM) and its task, called a target task. Then the application and selection procedures are conducted to build a team as shown in Fig. 1, where *fitness* and *resumé* are scalar values representing quantitative suitability of agents. The four-phase algorithm repeats until all tasks are assigned, and the detailed description for each phase is introduced in the next subsection. One complete series of the four phases is called one *round*.

In PCFA, the agent $i \in \mathcal{I}$, inherits the following local variables: the path list vector $\mathbf{p}^{(i)}$, the time table vector $\mathbf{t}^{(i)}$, the received application letter matrix $\mathbf{L}_{app}^{(i)}$, the received offer letter matrix $\mathbf{L}_{off}^{(i)}$, the position vector $\mathbf{U}_p^{(i)}$, average speed $\mathbf{U}_v^{(i)}$, and the winning advertisement vector $\mathbf{A}^{(i)}$. Table 1 summarizes the usages of local variables with examples.

On the other hand, the information of given tasks are defined as a structured variable $\mathbf{T}$, which is assumed to be updated from the mission control center. For all $k \in \mathcal{K}$, the task k is composed of six elements: $\mathbf{T}_p(k)$, $\mathbf{T}_a(k)$, $\mathbf{T}_m(k)$ $(=Z_k)$, $\mathbf{T}_w(k)$ $(=w_k)$, $\mathbf{T}_\lambda(k)$ $(=\lambda_k)$, and $\mathbf{T}_0(k)$ $(=t_k^0)$. The variable $\mathbf{T}_p(k)$ denotes the position where the task k should be conducted. The variable $\mathbf{T}_a(k) = 1$ if the task k is assigned to some agents and 0 otherwise.

2.3.2 Phase 1: Advertisement Preparation

At the first phase, the agent i prepares a winning advertisement vector $\mathbf{A}^{(i)}$ as summarized in Algorithm 1. To discover the most appropriate task for the agent i, *fitness* list $f^{(i)}$ is calculated for all unassigned tasks. The agent i then selects the task k^* with the highest value among $f^{(i)}$. The *fitness* of the agent i regarding the task k, i.e., $f^{(i)}(k)$, is defined as

$$f^{(i)}(k) = s_k(t_{ETA}) = w_k e^{-\lambda_k(t_{ETA}+t_w-t_k^0)} \tag{10}$$

Each agent does not know the path list $\mathbf{P}$ of the all agents, and therefore agents cannot calculate the

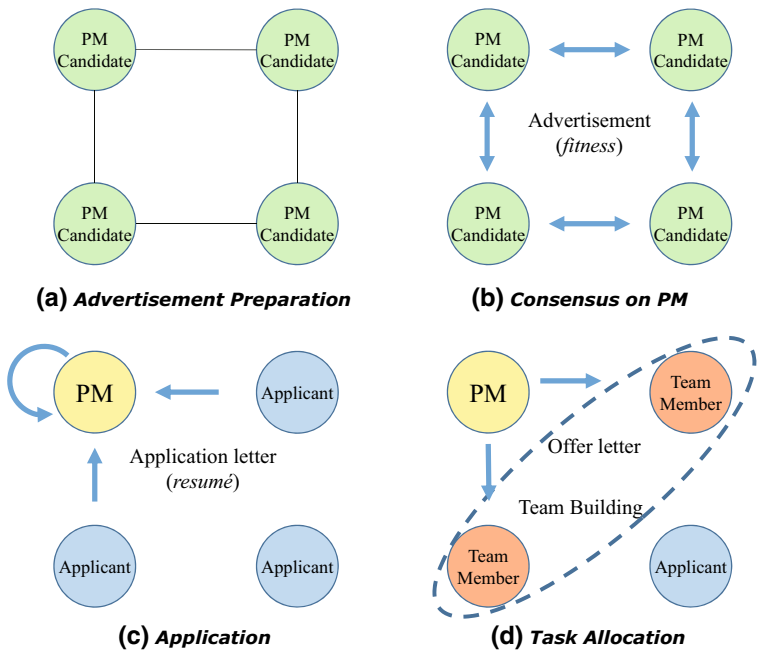


Fig. 1 Task allocation procedures in PCFA. Broadcasting message can be transferred when the network between the agents is connected, where solid line in (**a**) denotes the connectivity between two nodes. In this topology, direct communications between diagonal agents are not available

exact score function. Instead, the approximated score in Eq. 10 is utilized as an alternative in this study. Users may apply priorities between tasks by setting w_k and λ_k. When λ_k and t_k^0 are consistent for all tasks, the tasks with sufficiently large w_k will be performed earlier than other tasks. The diminishing rate of value can be adjusted by tuning λ_k. For example, the urgent task having top priority and short deadline can be defined as a task having high w_k and λ_k.

Note that the definition of ETA in Eq. 7 should be modified because the original definition of ETA is introduced to evaluate the objective function $J(\mathbf{P})$ on the premise that path list $\mathbf{P}$ is already determined. However, in the distributed TA process, when agent i calculates $t_{ETA}(i,k)$, agent i does not have task k; thus, $\mathbf{p}^{(i)}(1)$ cannot be equal to k. Additionally, the time t when t_{ETA} is calculated should be considered because evaluation of t_{ETA} may be necessary during the mission due to pop-up tasks. The modified definition of t_{ETA} for distributed TA procedure is as follows.

$$t_{ETA}(i,k,t) = \begin{cases} t + d_{ik}/v_i, & \text{if } \mathbf{p}^{(i)} = \emptyset \\ t_l + d_{lk}/v_i, & \text{otherwise} \end{cases} \tag{11}$$

Table 1 List of local variables of agent i

Variables	Example	Description
path list	$\mathbf{p}^{(i)} = [2\ 1]$	Visiting order of agent i: task 2→ task 1
time table	$\mathbf{t}^{(i)} = [100\ 200]$	Arrival times associated with $\mathbf{p}^{(i)}$
position	$\mathbf{U}_p^{(i)} = [10\ 100]$	Position of agent i: [10 100] (m)
average speed	$\mathbf{U}_v^{(i)} = 10$	Average speed of agent i: 10 m/s
application letter	$\mathbf{L}_{app}^{(i)} = \begin{bmatrix} j_1 & k & r_1 \\ j_2 & k & r_2 \end{bmatrix}$	Agent i knows that agents j_1 and j_2 applied to task k with *resumé* r_1 and r_2, respectively
offer letter	$\mathbf{L}_{off}^{(i)} = [j\ k\ t]$	Agent i knows that agent j sent an offer letter to agent i for task k with appointed time t
winning advertisement	$\mathbf{A}^{(i)} = [j\ k\ f]$	Agent i considers agent j as PM for task k with *fitness* f

In PCFA, advertisement for a certain task is only allowed to agents who have sufficient numbers of neighborhood for the task. Agent i's neighborhood, $\mathcal{N}^{(i)} \subset \mathcal{I}$, consists of agents connected with the agent i. In Fig. 1, each agent has two neighborhoods. When the agent i does not have sufficient neighborhood for a certain task, *fitness* for that task is zero (Algorithm 1 line 5). Note that $Z^{(i)}_{max}(= n(\mathcal{N}^{(i)}) + 1)$ is the maximum number of agents that can be mobilized by the agent i. This conditional statement restricts the candidate of team members to the neighborhood of the PM.

Additionally, the agent i computes the previous winning advertisement $\mathbf{A}^{(i)}_{prev}$, which is designed to be shared through communication between neighboring agents in the following phase. When the agent i fails to calculate the *fitness*, it generates a dummy winning advertisement (Algorithm 1, line 14).

Algorithm 1 Phase 1 for agent i

1: **procedure** ADVERTISEMENT PREPARATION(i, t)
2: $f^{(i)} = \mathbf{0}_{n(\mathcal{K})\times 1}$
3: $Z^{(i)}_{max} = n(\mathcal{N}^{(i)}) + 1$
4: **for** $k = 1 : n(\mathcal{K})$ **do**
5: **if** $(\mathbf{T}_a(k) = 0)$ & $(Z^{(i)}_{max} \geq Z_k)$ & $(n(\mathbf{p}^{(i)}) < L_{max})$ **then**
6: Calculate $f^{(i)}(k)$ ▷ Eq. (10)
7: **end if**
8: **end for**
9: **if** $\sum_{k=1}^{n(\mathcal{K})} f^{(i)}(k) \neq 0$ **then**
10: $k^* = \arg\max_{k\in\mathcal{K}} f^{(i)}(k)$
11: $q^* = f^{(i)}(k^*)$
12: $\mathbf{A}^{(i)} = [i, k^*, q^*]$
13: **else**
14: $\mathbf{A}^{(i)} = [i, 0, 0]$
15: **end if**
16: $\mathbf{A}^{(i)}_{prev} = \mathbf{A}^{(i)}$
17: **end procedure**

2.3.3 Phase 2: Consensus on PM

In this phase, the agent i makes effort to reach consensus on the PM and corresponding target task for the current round by negotiating with the neighboring agents as shown in Algorithm 2. Note that every agent prepares two advertisement vectors in phase 1, i.e., 1) previous winning advertisement vector $\mathbf{A}^{(i)}_{prev}$, and 2) winning advertisement vector $\mathbf{A}^{(i)}$. The first is broadcast to the neighboring agents, and the second is compared with the neighboring agents' previous winning advertisement vectors. If the *fitness* component of $\mathbf{A}^{(i)}$ is strictly less than the neighboring agent j's *fitness* component of $\mathbf{A}^{(j)}_{prev}$, $\mathbf{A}^{(i)}$ is replaced by $\mathbf{A}^{(j)}_{prev}$. Note that $\mathbf{A}^{(i)}_{prev}$ is updated by $\mathbf{A}^{(i)}$ before proceeding to the next iteration.

The above process is repeated until the PM is selected, and therefore, several iterations may be required during the phase. To consider the maximum number of the required iterations, let us consider the diameter of the network. The diameter is defined as the maximum distance of the two arbitrary vertices of the graph, where the distance is the length of the shortest walk between two vertices [20]. The agent i propagates the greatest *fitness* to the entire agents after comparing the *fitness* component with its neighboring agents. By a single broadcasting, the greatest *fitness* is propagated to the neighboring agents. Therefore, by definition, it can be inferred that the number of required iterations for the consensus does not exceed the diameter of the network topology. However, the decentralized agents may not be able to recognize the exact topology of the network because the communication connection between agents may change during the mission. Thus, the network diameter for the worst case, $N - 1$, is selected as a conservative limit (Algorithm 2 line 2). When the agents have information on the exact diameter of the network, $N - 1$ can be replaced by the network diameter.

Phase 2 is summarized in Algorithm 2∼3 where $\mathbf{A}^{(i)}(b)$ denotes the b-th element of $\mathbf{A}^{(i)}$; that is, $\mathbf{A}^{(i)}(2)$ is the task element and $\mathbf{A}^{(i)}(3)$ is the *fitness* element of $\mathbf{A}^{(i)}$. $\mathcal{N}^{(i)}(b)$ denotes the b-th element of $\mathcal{N}^{(i)}$. Note that tie-break rule is applied by prioritizing agent with a lower index (Algorithm 3 lines $7 \sim 9$).

Algorithm 2 Phase 2 for agent i

1: **procedure** CONSENSUS ON PM(i)
2: **for** $C^{(i)} = 1 : N - 1$ **do**
3: ConsensusPM(i)
4: **end for**
5: **end procedure**

Algorithm 3 ConsensusPM(i)

Broadcast $\mathbf{A}^{(i)}_{prev}$
for $u = 1 : n(\mathcal{N}^{(i)})$ **do**
  $j = \mathcal{N}^{(i)}(u)$
  if $\mathbf{A}^{(i)}(3) < \mathbf{A}^{(j)}_{prev}(3)$ **then**
    $\mathbf{A}^{(i)} = \mathbf{A}^{(j)}_{prev}$
  else if $\mathbf{A}^{(i)}(3) = \mathbf{A}^{(j)}_{prev}(3)$ **then**
    if $\mathbf{A}^{(i)}(1) > \mathbf{A}^{(j)}_{prev}(1)$ **then**
      $\mathbf{A}^{(i)} = \mathbf{A}^{(j)}_{prev}$
    end if
  end if
end for
$\mathbf{A}^{(i)}_{prev} = \mathbf{A}^{(i)}$

2.3.4 Phase 3: Application

As a result of phase 2, every agent knows the PM and its target task. In phase 3, each agent sends an application letter to PM. In PCFA, sending an application letter to the PM is allowed only for the agents directly connected to the PM. The *resumé*, which is included in the application letter, is defined as the ETA to the target task, i.e.,

$$r^{(i)}(k,t) = t_{ETA}(i,k,t). \tag{12}$$

On the other hand, a PM is not necessarily the most appropriate agent for the task because the qualification of the PM includes the number of its neighboring agents. When applicants are better than the PM, the role of the PM is only to recruit applicants by utilizing its networking ability as shown in Fig. 1. Thus, the PM should compete with other applicants to be a team member. Phase 3 is summarized in Algorithm 4, where the left arrow operator augments the right row vector into the left matrix.

Algorithm 4 Phase 3 for agent i

procedure APPLICATION (i,t)
  $j^* = \mathbf{A}^{(i)}(1)$
  $k^* = \mathbf{A}^{(i)}(2)$
  if $(i = j^*)$ **then**
    $\mathbf{L}^{(i)}_{app} \Leftarrow [i, k^*, r^{(i)}(k^*, t)]$
  else if $(Z_{k^*} > 1) \& (j^* \in \mathcal{N}^{(i)})$ **then**
    $\mathbf{L}^{(j^*)}_{app} \Leftarrow [i, k^*, r^{(i)}(k^*, t)]$
  end if
end procedure

2.3.5 Phase 4: Team Building

As a result of phase 3, PM has application letters from its neighboring agents. In phase 4, PM evaluates the suitability of applicants by comparing *resumé*, which is included in their application letters. Because the PM advertised a task that can be accomplished by itself and the neighboring agents, there always exists a sufficient number of applicants. The appointed arrival time is determined as the latest arrival time of the selected team members. Then, the PM sends offer letters to the selected team members to inform the appointed arrival time.

On the other hand, once the agent i receives the offer letter, it then augments the task and appointed time into its own path list $\mathbf{p}^{(i)}$ and time table $\mathbf{t}^{(i)}$, respectively. This team building procedure is summarized in Algorithm 5, where $\mathbf{p}^{(i)} \oplus_{end} \{k\}$ denotes that the task k is augmented at the end of the agent i's path vector $\mathbf{p}^{(i)}$.

Algorithm 5 Phase 4 for agent i

procedure TEAM BUILDING (i)
  $j^* = \mathbf{A}^{(i)}(1)$
  $k^* = \mathbf{A}^{(i)}(2)$
  if $(i = j^*)$ **then**
    Select Z_{k^*} applicants based on received *resumés*
    Determine appointed time t^* as latest time
    Send offer letters to selected applicants
    $\mathbf{T}_a(k^*) = 1$
  end if
  if Agent i received offer letter on task k^* **then**
    $\mathbf{p}^{(i)} = \mathbf{p}^{(i)} \oplus_{end} \{k^*\}$
    $\mathbf{t}^{(i)} = \mathbf{t}^{(i)} \oplus_{end} \{t^*\}$
  end if
  $\mathbf{L}^{(i)}_{app} = \mathbf{L}^{(i)}_{off} = \emptyset$
end procedure

2.4 Task-oriented Coalition Formation Algorithm: TCFA

In PCFA, a coalition is organized by the agreed-upon PM who is asked to rank applicants and send offer letters. This method can usually be utilized for cooperative timing missions even if the network is not fully

connected, but the capacity of multiple agents can be excessively limited due to the restriction that the team members should be directly connected with the PM. For instance, suppose that there exist four agents and a task, which should be conducted by the four agents. When the communication network of the four agents has a ring topology, as shown in Fig. 1a, the task cannot be accomplished because the maximum number of neighborhood is two. The motivation of TCFA is to handle this limitation; by relaying application letters, every agent can be a coalition member regardless of the network topology. In TCFA, agents make consensus on not only a PM and its target task but also team members by *additionally* sharing the information of applicants. Figure 2 shows the overall procedure of TCFA.

In phase 1, the condition on the number of neighborhood in PCFA is disregarded in TCFA when each agent prepares the advertisement. The remaining parts of phase 1 and phase 2 are identical to those of PCFA. In phase 3, the application letters are shared by each agent and their neighborhoods to reach a consensus on the letter. At the first broadcasting, every agent sends its application letter to neighboring agents, and agents augment the received letter to its own letter matrix. At the next broadcasting, every agent sends application letters accumulated from the previous broadcasting. In this way, for one broadcasting, all application letters are *radially propagated* from each agent (vertex) to its neighboring agents (adjacent vertices). From the fact that the diameter of the connected network is at most $N-1$, every application letter can be propagated through the whole nodes after $N-1$ broadcasts. Note that the information of the network topology was not utilized during phase 3.

However, agents do not have to share all application letters because only Z_k agents are required to perform task k. In addition, agents know that the PM is the most proper agent to task k, which means that $Z_k - 1$ members need to be selected through phase 3 and phase 4. Therefore, after receiving all application letters from neighboring agents, each agent keeps only high-scored $Z_k - 1$ application letters and deletes the others. By this manner, the amount of communication can be saved because Z_k is usually less than N. The modified phase 3 is summarized in Algorithm 6∼8. When the agents have identical *resumé* values, the agent with the lowest index is selected.

Algorithm 6 Modified Phase 3 for agent i

procedure APPLICATION (i,t)
  PrepareApp(i,t)
  ConsensusApp(i)
end procedure

Algorithm 7 PrepareApp(i,t)

$j^* = \mathbf{A}^{(i)}(1)$
$k^* = \mathbf{A}^{(i)}(2)$
if $(Z_{k^*} > 1)\&(i \neq j^*)$ **then**
  $\mathbf{L}_{app}^{(i)} = [i, k^*, r^{(i)}(k^*, t)]$
  $\mathbf{L}_{app,prev}^{(i)} = \mathbf{L}_{app}^{(i)}$
end if

Algorithm 8 ConsensusApp(i)

$k^* = \mathbf{A}^{(i)}(2)$
for $C^{(i)} = 1 : N-1$ **do**
  Broadcast $\mathbf{L}_{app,prev}^{(i)}$
  for $u = 1 : n(\mathcal{N}^{(i)})$ **do**
    $j = \mathcal{N}^{(i)}(u)$
    $\mathbf{L}_{app}^{(i)} \Leftarrow \mathbf{L}_{app,prev}^{(j)}$
    Delete duplicated letters in $\mathbf{L}_{app}^{(i)}$
  end for
  Maintain at most $Z_{k^*} - 1$ letters by comparing *resumé*
  $\mathbf{L}_{app,prev}^{(i)} = \mathbf{L}_{app}^{(i)}$
end for

In phase 4, every agent has the same awareness; i) the PM and its target task, and ii) $Z_k - 1$ most proper members. Therefore, the target task is allocated to the proper members and PM. Note that agents do not have to communicate with each other in phase 4 because all the necessary information for TA is already shared before phase 4. The modified phase 4 is summarized in Algorithm 9.

The aim of PM selection in TCFA is to choose the corresponding target task. By fixing the target task at phase 2, only the fitness for the target task needs to be shared, which reduces the communication overhead and computational load.

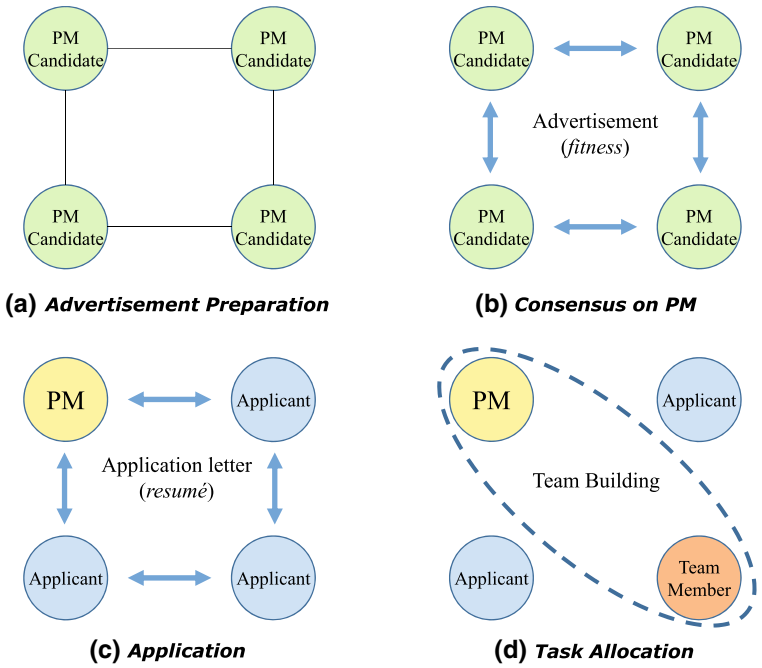


Fig. 2 Task allocation procedures in TCFA. Broadcasting message can be transferred when the network between the agents is connected, where solid line in (**a**) denotes the connectivity between two nodes. In this topology, direct communications between diagonal agents are not available

Algorithm 9 Modified Phase 4 for agent i

procedure TEAM BUILDING (i)
  $j^* = \mathbf{A}^{(i)}(1)$
  $k^* = \mathbf{A}^{(i)}(2)$
  $\mathbf{s}^*$ = indexes of applicants in $\mathbf{L}_{app}^{(i)}$
  if $i = j^*$ *or* $i \in \mathbf{s}^*$ **then**
    $\mathbf{p}^{(i)} = \mathbf{p}^{(i)} \oplus_{end} \{k^*\}$
    Determine appointed time t^* as latest time
    $\mathbf{t}^{(i)} = \mathbf{t}^{(i)} \oplus_{end} \{t^*\}$
    $\mathbf{T}_a(k^*) = 1$
  end if
  $\mathbf{L}_{app}^{(i)} = \mathbf{L}_{app,prev}^{(i)} = \emptyset$
end procedure

2.5 Greedy Distributed Allocation Protocol

For a comparative study, greedy distributed allocation protocol (GDAP) [45] is modified and adapted to our problem. The coalition formation scheme, including the advertising and applying processes of GDAP, is similar to the proposed TA algorithms, and different aspects will be treated in the subsequent section dealing with properties of the proposed algorithms.

The idea of GDAP is as follows. There are tasks that require a certain combination of resources, and all agents have some resources. The information of each task is randomly distributed to each agent at the beginning of the TA process, and these agents are called the managers of each task. When the network topology between agents is not fully connected, only the manager's neighboring agents are permitted to contribute to the task. In other words, not all agents get the information about all tasks. All agents, including managers, are called contractors. Each manager finds contractors who will help with their tasks, and each contractor makes bids to the manager who has the highest efficiency among the contractor's neighboring managers. If sufficient resources are supplied by contractors, the manager selects a set of contractors randomly. If a manager fails to build a coalition for a certain task, then the task is deleted from the manager's task list.

To apply GDAP to our problem, GDAP is modified as follows. In phase 1, the agent i, who is a manager of several tasks, calculates its *fitness* defined in Eq. 10 for the tasks distributed to the agent i. In other words, manager agent i selects the most suitable task by choosing the highest value among $f^{(i)}$. Manager agent i then advertises itself to its neighborhood via

$\mathbf{A}^{(i)}$ defined in Table 1. If $Z_{max}^{(i)}$ is less than the number of required agents for the task, the agent i hands over the task to one of its neighboring agents instead of deleting it. This is done by transmitting the task information to one of its neighboring agents and removing the information from agent i itself. In phase 2, the agent i applies to the agent who advertised the highest *fitness* as in PCFA and TCFA. In phase 3, if a manager has sufficient applicants, the manager selects the best team members according to the *resumé* value. In our implementation, each manager is not a team member of its tasks by default, but it should compete with other applicants because the arrival time of the manager is not always shorter than that of others. From the perspective of this study, GDAP consists of three phases, as shown in Fig. 3.

Note that GDAP requires less communication and no iterations due to the lack of consensus process. In dynamic environment, rapid decision making may enhance the TA efficiency. Also, in decentralized TA algorithm, there exists a trade-off between the communication/computation effort and the performance. In this context, the modified GDAP is worthwhile to be compared with the proposed algorithms.

3 Properties of TA Algorithm

3.1 Convergence

In this section, it is proven that a conflict-free solution generated by the proposed TA algorithms converges. *Convergence* of a TA algorithm denotes that the algorithm is capable of allocating given tasks to the agents within a finite time. The *conflict-free* represents that the resultant path list is feasible with respect to the constraints in Eqs. 2∼4. The convergence of PCFA in a connected network is guaranteed by Theorem 1.

Theorem 1 *Consider PCFA and given tasks with involved agents. Let us assume that the maximum number of necessary agents for the given tasks is bounded by the number of maximum available agents* $Z_{max}^{(i)}$ $\forall$ $i \in \mathcal{I}$ *and that the network topology is connected but not necessarily fully connected. Then, within a finite time, PCFA converges to a conflict-free solution.*

Proof The theorem can be proven by checking the possible bottlenecks of the four phases. In phase 1, each agent computes its own *fitness* and builds a winning advertisement vector, and therefore, no bottleneck exists. In phase 2, when the network is connected, the number of iterations required to reach a consensus on the PM is bounded above by $N - 1$. When the *fitnesses* of different PM candidates are the same, the candidate with lower agent index is selected. This procedure is applied similarly when the *resumés* of different applicants are equal. In phase 3, the application does not produce a bottleneck. In phase 4, because the agents advertised a task requiring themselves and their neighboring agents, the PM always has a sufficient number of applicants. Therefore, within a finite time, a single task will be allocated to a group of agents during a round.

Because it is assumed that the maximum number of necessary agents for the given tasks is bounded by

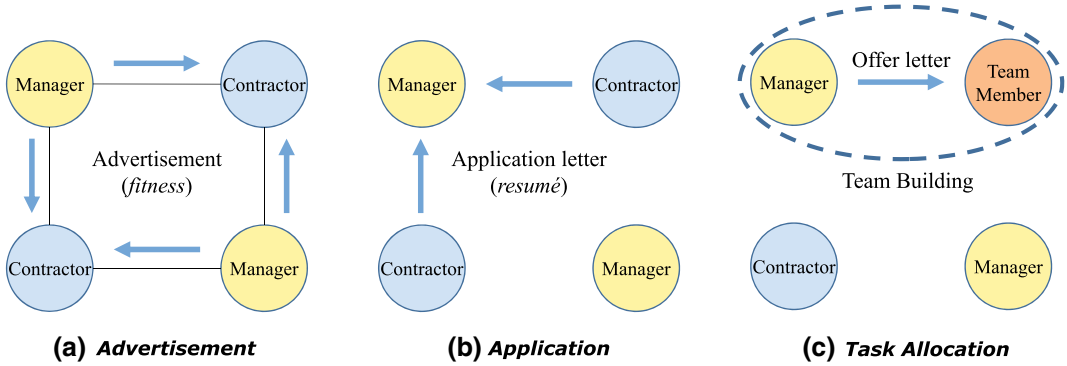


Fig. 3 Task allocation procedures in GDAP. Broadcasting message can be transferred when the network between the agents is connected, where solid line in (**a**) denotes the connectivity between two nodes. In this topology, direct communications between diagonal agents are not available

the number of maximum available agents, within a finite time, M tasks will be assigned during M rounds. Moreover, because each task is augmented at the end of the existing path vector, the resulting path vector $\mathbf{P}$ satisfies the DAG constraint in Eq. 4. □

According to Theorem 1, the convergence of TCFA is straightforward. Phase 1 and phase 2 of TCFA do not form a bottleneck, as in the proof of Theorem 1. In phase 3, at most, $N-1$ iterations are required to reach a consensus on the application letter. Finally, in phase 4, because every agent recognizes the PM and the team members, a single task is allocated to one team at a round.

3.2 Scalability

3.2.1 Amount of Communication

The amount of communication for TA is analyzed in this section. As shown in Lemma 1, for a connected network, there exists a polynomial upper bound for the number of communications required when allocating given tasks. It is assumed that the agents are not informed of the diameter of the current network. Additionally, one broadcast of an agent to its neighborhood is counted as one communication. Therefore, it can be stated that PCFA is scalable to large problems with regard to the amount of communication.

Lemma 1 *For completing the TA process using the PCFA, where $N \leq 256$, upper bounds of the number of communications and the total communication overhead in terms of bytes can be computed as $C_{TA}(N, M)$ and $B_{TA}(N, M)$, respectively.*

$$C_{TA}(N, M) = 2M(N-1) \tag{13}$$

$$B_{TA}(N, M) = 12M(N-1) \tag{14}$$

Proof By Theorem 1, M tasks are allocated within M rounds. Now, let us consider the maximum number of communications for each round. In phase 1, no communication is required. In the worst case of phase 2, each agent broadcasts its winning advertisement $\mathbf{A}^{(i)} = [j, k, f]$ for $N-1$ times during $N-1$ iterations. In phase 3, each agent sends at most one application letter $\mathbf{L}_{app}^{(i)} = [j, k, r]$ to the PM. In phase 4, the PM sends at most $N-1$ offer letters $\mathbf{L}_{off}^{(i)} = [j, k, t]$ to its neighboring agents. Note that the communication is required in either phase 3 or phase 4 because the PM does not send an application letter to other agents. Therefore, the number of communications for allocating a task is bounded above by $2(N-1)$, and for M tasks, the upper bound is $2M(N-1)$. On the other hand, six bytes are uniformly required for each communication of $\mathbf{A}^{(i)}$, $\mathbf{L}_{app}^{(i)}$, and $\mathbf{L}_{off}^{(i)}$; two bytes for two natural numbers $j, k \leq 256$, and four bytes for real numbers f, r, t with single-precision. Thus, the maximum overhead in terms of bytes are $12(N-1)$ bytes. □

The number of communications required for TCFA can be computed as in Lemma 1. For TCFA, communication is required only in phase 2 and phase 3, and the maximum number of communications in phase 2 is $N-1$, which is identical to that of PCFA. Additionally, phase 3 requires at most $N-1$ communications for consensus on application letters. Therefore, the maximum number of communications for TCFA is identical to that of PCFA. However, the total communication overhead of TCFA is greater than that of PCFA, because $\mathbf{L}_{app}$ is a $(Z_k - 1) \times 3$ matrix in TCFA while $\mathbf{L}_{app}$ is a 1×3 vector in PCFA. For the target task k, the communication overhead in terms of bytes using TCFA is $6Z_k(N-1)$ bytes; $6(N-1)$ bytes in phase 2 and $6(Z_k - 1)(N-1)$ bytes in phase 3.

3.2.2 Time Complexity

The proposed algorithms are scalable to large-sized problems in terms of time complexity. Theorem 2 shows that PCFA runs in a polynomial time.

Theorem 2 *The asymptotic worst-case time complexity of the PCFA with M tasks and N agents can be expressed as follows.*

$$T_{TA}(N, M) = O(M^2 + MN^2 + MN\log(N)). \tag{15}$$

Proof By Theorem 1, M tasks are allocated to N agents within M rounds.

Now, let us consider the time complexity of each round. In phase 1, the time complexity of the first *for statement* in Algorithm 1 line 4 is $O(M)$ because the ETA is calculated for M times. The *if-statement* in line 9 of Algorithm 1 requires $O(2M)$ time complexity.

For the worst case of phase 2, each agent compares *fitness* with the $N-1$ neighboring agents during $N-1$ iterations. Thus, it can be concluded that the phase 2 has a time complexity of $O((N-1)^2)$. Phase 3 has a constant time complexity, and this means that the number of maximum elementary operations in phase 3 does not depend on the number of the involved agents and given tasks. In phase 4, at most, N elements are sorted, and it is well known that the time complexity of the sorting is $O(n\log(n))$, where n is the number of elements to sort [10]. Therefore, the time complexity of phase 4 is $O(N\log(N))$. By summing up the aforementioned numbers, the worst-case time complexity of each round can be described as $O(3M+(N-1)^2+N\log(N))$. Moreover, the asymptotic worst-case time complexity can be expressed as $O(M+N^2+N\log(N))$. Finally, for M tasks, the time complexity of PCFA can be expressed as $O(M^2+MN^2+MN\log(N))$. □

The time complexity of TCFA can be determined as in Theorem 2. The time complexities of phase 1 and phase 2 of TCFA are the same as those of PCFA, and, at most, $N-1$ iterations are required for phase 3 of TCFA. During each iteration, the application letters collected by the agent i from its neighboring agents should not be duplicated. For deleting duplicated letters of agent i and agent j, at most, $(N-1)^2$ comparisons are required. Because agent i has at most $N-1$ neighboring agents, no more than $(N-1)^3$ comparisons are necessary. After deleting the duplicated application letters, a comparison sort is performed, and this process requires $O((N-1)\log(N-1))$. Therefore, it can be stated that the phase 3 has a time complexity of $O((N-1)^4+(N-1)^2\log(N-1))$. For phase 4, the time complexity of $O(N-1)$ is demanded to check the acceptance. Thus, the asymptotic worst-case time complexity of TCFA can be described as

$$T_{TA}(N,M) = O(M^2+MN+MN^2+MN^2\log(N)+MN^4). \tag{16}$$

3.3 Performance

The performance of the proposed TA algorithms is described in this section. A drawback of the proposed algorithms is that the optimal solution may not be obtained because the TA problem is addressed in a decentralized manner. However, the proposed algorithms have several merits. First, the proposed algorithms are applicable for various types of network topologies within a connected network. Specifically, for a connected network, the adjacency matrix of the network topology does not have to be shared by all of the agents. Therefore, the agents require only the indices of the neighboring agents, and this information can be easily obtained by the ping test.

Second, the algorithms induce less computational and communicational burden because the necessary calculations are composed of fundamental arithmetic operations or logical operations (i.e., comparison). The number of communications and total required overhead are upper bounded by a polynomial.

Third, the proposed algorithms can be extended to various cooperative TA problems where the *fitness* and *resumé* can be defined. Because this study focuses on tasks with cooperative timing constraints, the TA algorithm only decides the sequence of the path list and the time table. Proof of convergence can be equivalently applied regardless of the definition of *fitness* and *resumé*.

3.4 Comparison with GDAP

There is an important difference between the proposed algorithms and GDAP. While the selection of the auctioneer is negotiated for each round in the proposed methods, all auctioneers are chosen randomly in GDAP. As a result, TA solutions are different even in the fully connected network. In other words, every agent using the proposed algorithms calculates its *fitness* for all tasks to be a PM. In GDAP, however, only the manager agents calculate their *fitness* for the tasks allowed to them.

In the resource management problem, which is the target problem of GDAP, the choice of an auctioneer is not an important factor. However, in the TA problem considering mission completion time, the choice of an auctioneer can improve the efficiency. For example, suppose a dynamic environment has several disconnected subnetworks with a limited communication range. In this case, if the manager of a certain task is too far from the corresponding task, GDAP will form an inefficient coalition.

GDAP has strong points with respect to less rounds and communication, and it can allocate multiple tasks

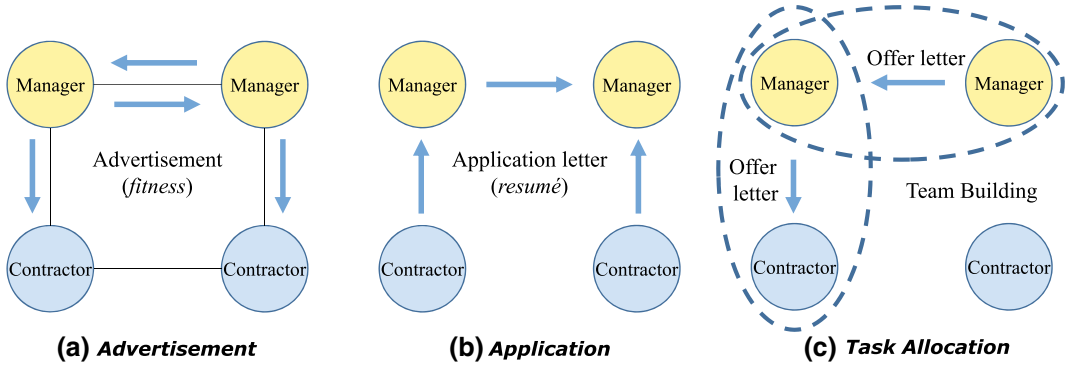


Fig. 4 Possible conflict in GDAP. Broadcasting message can be transferred when the network between the agents is connected, where solid line in (**a**) denotes the connectivity between two nodes. In this topology, direct communications between diagonal agents are not available

during one round even in the connected network. However, a conflict can occur in GDAP when the network is not *fully-connected*, which means all pairs of vertices are directly connected, because managers are concurrently contractors. For example, as in the case shown in Fig. 4, suppose a manager broadcasts an advertisement of itself in phase 1. Then, the manager may apply to the neighboring manager with higher *fitness* in phase 2. In phase 3, there are sufficient agents applied to the manager, and the manager also becomes a member of the team. In this case, if the manager is also selected by another manager, then the manager is doubly assigned to two different teams. To resolve this conflict, more cross-checking is necessary.

4 TA Algorithm in Dynamic Environment

4.1 Challenges in Dynamic Environments

In dynamic environments, additional tasks may be given to agents. Assuming that network connectivity depends on the relative distance between agents, the network topology will change dynamically or may even be disconnected due to the mobility of the agents. Therefore, the proposed algorithms may not work in dynamic environments as intended because a static, connected network during each TA round is assumed. Specifically, a disconnection during consensus progress causes a conflict. For example, two disconnected subgroups may have different ideas about who the PM is. Additionally, a disconnection during the application phase creates a disagreement about who the team member is.

In fact, conflicts over a disconnected network are inevitable when a decentralized TA algorithm is used, especially for a strongly coupled problem such as the coalition formation. The major issue is how to minimize performance degradation over the disconnected network.

4.2 Decentralized TA Architecture in Dynamic Environments

This section addresses an extension of the proposed TA algorithms to the problem in dynamic environments. The assumptions for this extension are as follows: i) there is a mission control center that monitors

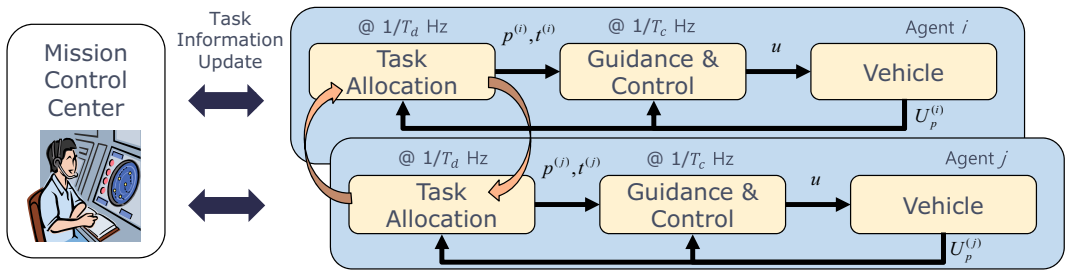


Fig. 5 Real-time decentralized TA architecture

all of the agents and updates the task information **T** mutually, ii) the clock is synchronized, and iii) each agent knows the list of agents in its subnetwork.

Let us consider that each agent has TA block for a high-level controller and guidance and a control block for a low-level controller. Once TA block calculates the path list and corresponding time table, the low-level controller causes the vehicle to arrive at the target in time. Guidance and control block run every T_c sec, and TA block runs every T_d sec, where T_d depends on communication bandwidth. Figure 5 shows the architecture of the proposed TA algorithm for real-time implementation.

TA block consists of four phases, and one of four phases is performed at each execution of TA block. At the phases requiring several iterations for consensus with neighboring agents, only one iteration is performed at each execution of TA block. For instance, it takes $(N-1)T_d$ sec for phase 2 of PCFA. In this study, the phase token $K^{(i)}$ is adopted, which indicates the number of phase to be executed. The agent i resets the phase token $K^{(i)}$ to one in two cases. First, if there are no unassigned tasks, agents do not proceed with the TA process and reset the phase token to one. Another case is when members of agent i's subnetwork change during the mission. Note that the proposed algorithm requires synchronous phase scheduling in each subnetwork, and therefore, all members of the subnetwork should have the same value of phase token.

In phase 2, consensus of the PM requires at most $n_s^{(i)} - 1$ iterations where $n_s^{(i)}$ is the number of nodes in the agent i's subnetwork. Still, the information of the network topology is not used. In phase 3, the *resumé* should be changed because there exists a time gap between application time and team building time. Simultaneous arrival may fail if this gap is neglected. The modified *resumé* of PCFA is as follows.

$$r^{(i)}(k,t) = t_{ETA}(i,k,t) + 2T_d \tag{17}$$

where $2T_d$ compensates for the time gap as well as the travelled distance of the agent i during T_d sec. The entire process of PCFA dealing with dynamic environments is presented in Algorithm 10.

Similarly, TCFA can be extended for dynamic environments. Agents prepare the application letter in the first run of phase 3 and then begin to make a consensus in the second run. The modified *resumé* of TCFA is

$$r^{(i)}(k,t) = t_{ETA}(i,k,t) + 2(n_s^{(i)} - 1)T_d \tag{18}$$

where $2(n_s^{(i)} - 1)T_d$ compensates for the time gap as well as the travelled distance of agent i during $(n_s^{(i)} - 1)T_d$ sec. The modified pseudocode of case 3 for TCFA is presented in Algorithm 11 where $F_3^{(i)}$ is initialized to zero at case 1 of TCFA.

Algorithm 10 TA Block for PCFA (i)

```
if There exist unassigned tasks then
    n_s^(i) = number of nodes of subnetwork
    Update location of RP
    if nodes of subnetwork are changed then
        K^(i) = 1
        L_app^(i) = ∅
    end if
    switch K^(i) do
        case 1
            Advertisement Preparation(i,t)
            K^(i) = 2
            F_2^(i) = 0
        case 2
            ConsensusPM(i)
            F_2^(i) = F_2^(i) + 1
            if F_2^(i) = n_s^(i) − 1 then
                K^(i) = 3
            end if
        case 3
            Application(i,t)
            K^(i) = 4
        case 4
            TeamBuilding(i)
            K^(i) = 1
else
    K^(i) = 1
end if
```

Algorithm 11 Case 3 for TCFA

```
case 3
    if F_3^(i) = 0 then
        PrepareApp(i)
    else
        ConsensusApp(i)
    end if
    F_3^(i) = F_3^(i) + 1
    if F_3^(i) = n_s^(i) then
        K^(i) = 4
    end if
```

4.3 Rally Point

In the ST-MR problem, multiple agents form a coalition to perform a common task, and communication between agents is required for negotiating which agents will be included in the coalition. In dynamic environments, however, the number of members in the subnetwork may be insufficient to perform the task due to a limited communication range. This problem can be resolved by adopting the *rally point* (RP) which is the designated place when an agent does not have any tasks to perform. Agents around the RP become connected, and thus, they can proceed in the TA process.

The adaptive positioning of the RP would be better than the stationary RP in many applications. For instance, in a friendly region, the geometric center of uncompleted tasks can be a time-efficient choice. During the SEAD mission, the preferable location of RP is on the safe border of the dangerous region and concurrently close to the base.

The adaptation law should generate the same location of RP through all agents without using other agents' positions because the precise position information of other agents is hard to update continuously. To derive the adaptation law for the RP position in SEAD mission, convex hull and Minkowski sum [13] are utilized. In the algorithm, the disk representing the surface to air missile (SAM) radar, which is the uncompleted task, is approximated as a hexagon, and the node points are made up of vertices of uncompleted tasks. Now, the convex hull of the node points becomes the boundary points of the dangerous region. Considering safe distance from the dangerous region, the Minkowski sum of the convex hull with a loitering circle is calculated. Finally, as shown in Fig. 6, the closest point from the base among the convex hull of the Minkowski sum is selected as the RP.

4.4 Convergence

In this subsection, convergence of the proposed algorithms in dynamic environment is analyzed. Note that the term *convergence* means that the all tasks can be allocated to the agents within a finite time. Let us remind the assumptions in Section 2.2; agents are allowed to communicate with each other synchronously and the network should be static and connected. For the synchronous communication, the phase token and its update rules are adopted, which reset the token when members of subnetwork are changed. For the static and connected network, RP was introduced. By gathering the agents not having any tasks around the RP, the static and connected network can be expected near the RP. Therefore, according to Theorem 1, convergence of the proposed algorithms are guaranteed in the dynamic environment.

4.5 Deletion of Duplicated Allocation

Disconnected subnetworks due to the limited communication range may yield duplicated allocations because each subnetwork does not have the information of the other subnetworks. In this study, it is assumed that mission control center resolves this problem by noticing the status of completed tasks to the agents. When the mission control center receives the completion notice reported by the coalition who visited the task, then it broadcasts the completeness of tasks as shown in Fig. 5. Note that this treatment may degrade the performance because multiple coalitions may head to the same task until one coalition completes the task.

5 Numerical Results

Numerical simulations are carried out to demonstrate the performance of the proposed TA algorithms. The scalability in a connected network is verified via a Monte Carlo simulation. Additionally, the proposed algorithms are applied to the dynamic SEAD scenario, a primary application of this study. The simulation environment includes a personal computer equipped with an Intel Core i5-4670 @ 3.40 GHz with 16 GB of RAM, and MATLAB on Windows 7 operating system is chosen as a numerical simulation tool.

5.1 Scalability

The scalability of the proposed algorithms is examined for a problem in a static, connected network. By Theorem 2, the proposed TA algorithms have polynomial time complexity, and therefore, the parallel runtime, which is obtained by dividing the total runtime by N [3], can be estimated by the time complexity formula. The estimated parallel runtime $\hat{t}_r$ can be obtained by

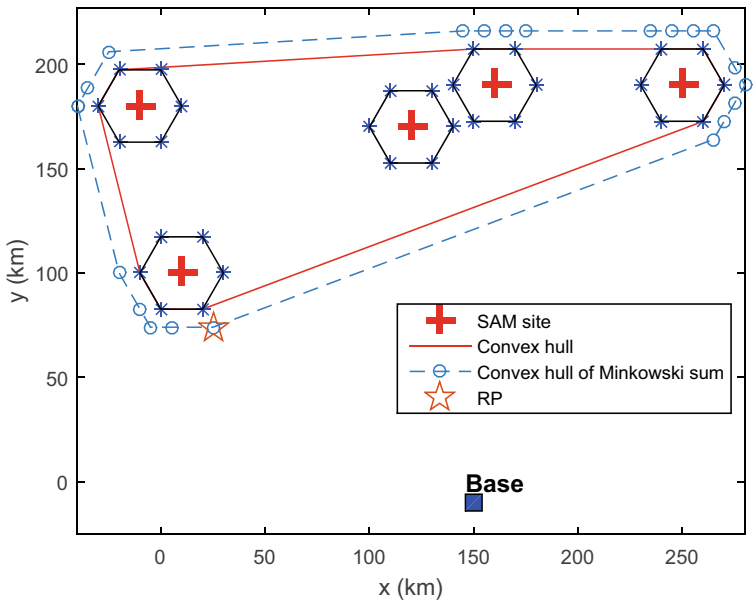


Fig. 6 Choice of rally point

adapting two unknown parameters for the simplified time complexity formula as

$$\hat{t}_r^c = \alpha_c M^2 + \beta_c M N^2, \tag{19}$$

$$\hat{t}_r^d = \alpha_d M^2 + \beta_d M N^2 \log(N), \tag{20}$$

where α_c and β_c are unknown estimator parameters of PCFA, and α_d and β_d are those of TCFA. The estimator parameters are identified by the least squares method, and Monte Carlo simulations are carried out to obtain the data for identification.

The runtime for various sizes of the problems are obtained by considering M tasks and N agents where $1 \leq M \leq 30$ and $1 \leq N \leq 30$, resulting in 900 different problem sizes. In addition, 100 Monte Carlo simulations are performed for each problem size so that 90,000 different problem cases are generated. The initial positions of the agents and the tasks are randomly generated within a 300 km by 200 km area, and the number of UAVs required for each task Z_k is randomly chosen between 1 to $Z_{max} = \max_{i \in \mathcal{I}} Z_{max}^{(i)}$, where $Z_{max}^{(i)} \equiv n(\mathcal{N}^{(i)}) + 1$ for phase 1. Thus, it can be stated that the network topology determines Z_{max}. The random walk approach, which generates a connected network by connecting two random vertices with an edge, is used for each simulation. The graph connectivity is determined by examining the Laplacian matrix. The graph is determined to be connected [20] when the second smallest eigenvalue of the Laplacian matrix is greater than zero. For a more general random network, N_r pairs of random vertices are connected with edges after the graph is connected, where N_r is randomly selected between 1 and 30.

For each problem size, the maximum parallel runtime t_r of the 100 Monte Carlo simulations was collected as the worst-case value. The 900 sets of the (N, M, t_r) were used to identify the unknown parameters, and the goodness of fit [32] was evaluated by using the normalized mean square error R^2.

The identification results are summarized in Table 2, and Fig. 7a shows a comparison between $\hat{t}_r$ and t_r for each problem size. The estimated runtime shows excellent agreement with the worst-case runtime with the exception of several outliers. From Fig. 7a, it can be concluded that the estimator reasonably represents the worst-case runtime. The difference between the estimated runtime and the worst-case runtime of PCFA and TCFA are shown in Figs. 7b and 8, respectively.

The polynomial time complexity is validated from the fitting results for a specific problem size by evaluating R^2 and the comparison graph. The simulation results show that the proposed TA algorithms can solve large-size problems, for example, the case when N and M are both 30, within one second. However, in the simulation results, network bandwidth was not considered, and therefore, the presented runtime can

Table 2 Estimated parallel runtime

Formula	$\hat{t}_r^c = \alpha_c M^2 + \beta_c MN^2$		$\hat{t}_r^d = \alpha_d M^2 + \beta_d MN^2 \log(N)$	
Parameters	α_c	$2.7833 * 10^{-5}$	α_d	$1.9975 * 10^{-4}$
	β_c	$3.2584 * 10^{-6}$	β_d	$1.0549 * 10^{-5}$
Goodness of Fit	R^2	0.9671	R^2	0.9356

be considered an ideal lower bound for a practical application.

The number of communications for the 100 Monte Carlo simulations is also compared with the communication bound stated in Lemma 1. To pick the worst case, the maximum number of communications per agent is logged during the Monte Carlo simulations. Figure 9 shows the maximum communications with upper bound with respect to the problem size $N(=M)$. As derived in Eq. 13, the number of communications grows quadratically with the problem size. PCFA has margins from the upper bound because the number of offer letters is generally less than $N-1$. On the other hand, TCFA communicates exactly the same as the bound because $N-1$ iterations are performed in phase 2 and phase 3, as the network topology is assumed to be unknown.

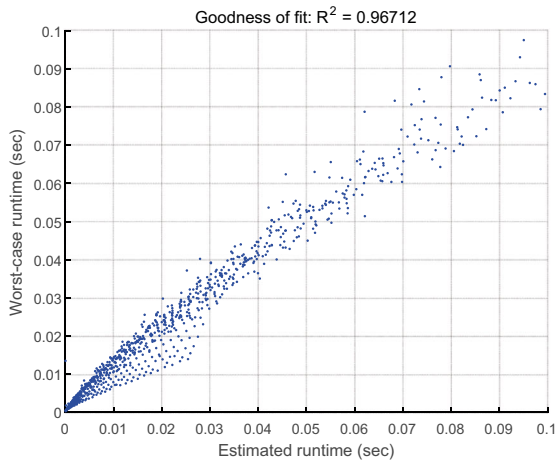
5.2 Application: SEAD Scenario

5.2.1 SEAD Environment

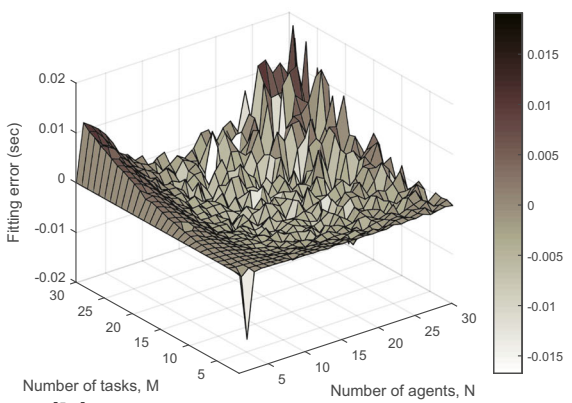
Figure 10a shows the two-dimensional battlefield considered for the SEAD mission, of which the objective is the complete destruction of the entire targets, i.e., surface-to-air missiles (SAMs), as soon as possible. Because SAMs are very dangerous, they must be simultaneously attacked by multiple UAVs. In Fig. 10, the number inside the parentheses of the task represents the number of required UAVs, which implies the degree of risk. The solid line connecting two UAVs means that those UAVs are within communication range. A UAV is considered as a point mass, and the collision between UAVs is neglected. The speed of the UAV is set to 200 m/sec. Figure 10b shows the dynamic environment at 800 sec, where T6 and T7 are pop-up tasks and T1 and T3 are completed tasks. The dotted line connecting a UAV and a task is the remaining path of the UAV.

5.2.2 Path Planning for SEAD

The simultaneous arrival strategy for the SEAD mission is described in this subsection. Let us remind that each UAV has its own path list and corresponding time table. The proposed algorithms augment the newly allocated task at the end of the path list. In addition, the appointed time is decided as the latest time of

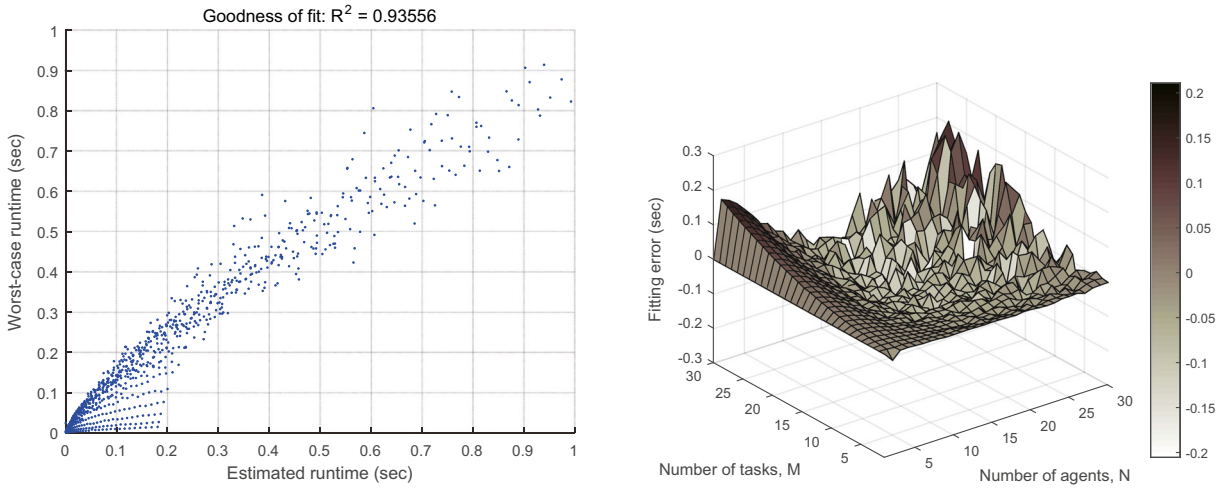


(a) Runtime comparison



(b) Estimation error of the worst-case runtime

Fig. 7 Parallel runtime estimation (PCFA)



(a) Runtime comparison

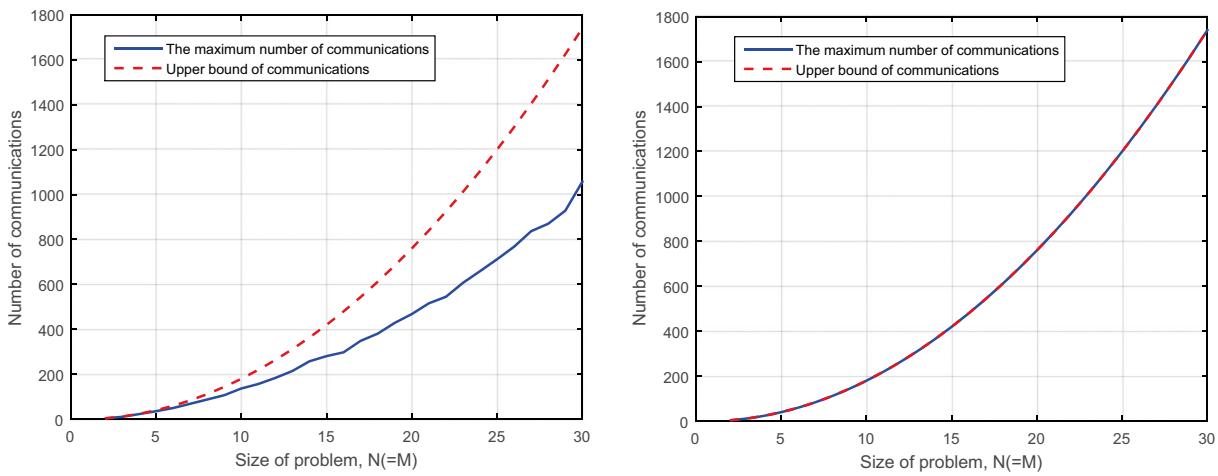
(b) Estimation error of the worst-case runtime

Fig. 8 Parallel runtime estimation (TCFA)

the team; thus, the scheduled time table and path list cause the UAVs to arrive at the common task simultaneously. When a UAV arrives earlier, it loiters around the task at a radius R_{safe}, where R_{safe} is the radius of the loitering pattern for the task position. When the estimated time to the task is same as the appointed time, UAVs steer their way to the task. Because the working time at the target is relatively short for the SEAD mission, t_w is set to zero in this study. After the completion of the task, each UAV advances to the next task. A UAV returns to the RP when all the tasks assigned to it are completed. If every UAV is loitering around the RP after finishing the tasks of the mission, they return to the base.

5.2.3 TA Results in Dynamic Environments with Various Communication Ranges

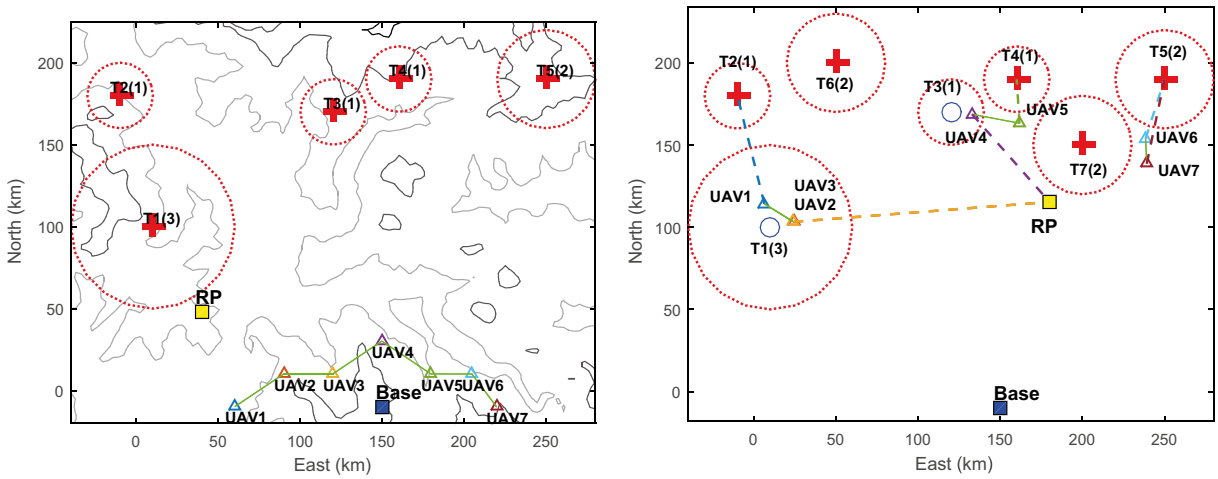
In this section, the performance of the proposed algorithms with various communication ranges is analyzed through Monte Carlo simulations. Prior to that, the detailed TA progress by TCFA is shown in Table 3 as a sample scenario where the description on each variable was introduced in Table 1. The scenario is



(a) PCFA

(b) TCFA

Fig. 9 Effect of problem size on amount of communication per agent



(a) Environment at 0 sec **(b) Environment at 800 sec**

Fig. 10 SEAD environment

selected as in Fig. 10b where the communication range is 40 km and the period of TA block T_d is 1 sec. According to the definition of *fitness* in Eq. 10, the inherent worth of task k, w_k, denotes the priority of the task k. In our simulations, however, the difference of priority among tasks was not considered, and w_k is set as 100 denoting the maximum score in our simulation for convenience. Time-discounting factor λ_k is set as 0.001 to reduce the score to '$1/e$' of w_k exponentially as 1,000 seconds have passed from the occurrence of the task.

At 800 sec, each UAV receives the information of T6 and T7 from the mission control center, thus making an advertisement of themselves. The waypoint number 100 denotes the RP. There are three disconnected subnetworks; group 1 consists of UAV1, UAV2, and UAV3; group 2 consists of UAV4 and UAV5; and group 3 consists of UAV6 and UAV7. At 801 sec, each subnetwork tries to make a consensus on the PM. Note that the numbers of required iterations for each group are 2, 1, and 1, respectively (Algorithm 10 line 16). Group 2 and 3 agree that the PM is UAV4 and UAV6, respectively. At 802 sec, group 1 makes a consensus on the PM as UAV3, while group 2 and 3 prepare application letters for consensus in phase 3 (Algorithm 11 line 3). At 803 sec, group 2 and 3 now make a consensus on the application letter, while UAV1 and UAV2 in group 1 prepare application letters. At 804 sec, members in group 2 and 3 allocate T7 simultaneously, while members in group 1 now start to make a consensus on the application letter. As in this case, duplicated allocations can occur in the dynamic environment due to the limited communication range. This cannot be avoided without more communication between the mission control center and the UAVs. In this study, duplicated allocations are resolved by deleting completed tasks in the path list and the time table based on the assumption that completeness of tasks is updated from the mission control center. At 805 sec, group 1 makes a consensus on the application letter, while group 2 and 3 intend to begin the TA process for T6. At 806 sec, group 1 allocates T6. At 807 sec, phase tokens for all UAVs are reset to one and stop the TA process because all tasks are assigned (Algorithm 10 line 26).

For Monte Carlo simulations, 100 random scenarios are generated. For each random scenario, 10 tasks and 10 agents are considered, and initial positions of tasks are randomly determined within a 300 km by 200 km area without overlapping on each other's perimeters while agents are located around the base. The number of required agents for each task, Z_k, is randomly chosen between one to five, and R_{safe} of each task is chosen between 20 km to 50 km proportional to Z_k. The maximum number of allowable tasks for each agent L_{max} is set as 5. Among 10 tasks, the information regarding two tasks, the other two tasks, and the last task is disseminated to agents at 800, 1,500, and 1,800 sec, respectively. On the other hand, the eight communication ranges considered are 20, 30, 50,

Table 3 Detailed TA Progress. Variable definitions are summarized in Table 1

Time (sec)	i	$K^{(i)}$	$\mathbf{A}^{(i)} = [j, k, f]$	$\mathbf{L}_{app}^{(i)} = [j, k, r]$	$\mathbf{p}^{(i)} = [p_1, ...]$	$\mathbf{t}^{(i)} = [t_1, ...]$
800	1	1	[1,6,51.925]	[ ]	[100,1,2]	[0,726,1139]
	2	1	[2,6,60.583]	[ ]	[100,1,100]	[0,726,1026]
	3	1	[3,6,60.599]	[ ]	[100,1,100]	[0,726,1026]
	4	1	[4,7,70.515]	[ ]	[100,3,100]	[0,734,884]
	5	1	[5,7,65.891]	[ ]	[100,4]	[0,934]
	6	1	[6,7,55.31]	[ ]	[100,5]	[0,1072]
	7	1	[7,7,55.31]	[ ]	[100,5]	[0,1072]
801	1	2	[3,6,60.599]	unchanged	unchanged	unchanged
	2	2	[3,6,60.599]			
	3	2	[3,6,60.599]			
	4	2	[4,7,70.515]			
	5	2	[4,7,70.515]			
	6	2	[6,7,55.31]			
	7	2	[6,7,55.31]			
802	1	2	unchanged	[ ]	unchanged	unchanged
	2	2		[ ]		
	3	2		[ ]		
	4	3		[ ]		
	5	3		[5,7,1223]		
	6	3		[ ]		
	7	3		[7,7,1398]		
803	1	3	unchanged	[1,6,1463]	unchanged	unchanged
	2	3		[2,6,1311]		
	3	3		[ ]		
	4	3		[5,7,1223]		
	5	3		[5,7,1223]		
	6	3		[7,7,1398]		
	7	3		[7,7,1398]		
804	1	3	unchanged	[2,6,1311]	[100,1,2]	[0,726,1139]
	2	3		[2,6,1311]	[100,1,100]	[0,726,1026]
	3	3		[2,6,1311]	[100,1,100]	[0,726,1026]
	4	4		[ ]	[100,3,100,7]	[0,734,884,1223]
	5	4		[ ]	[100,4,7]	[0,934,1223]
	6	4		[ ]	[100,5,7]	[0,1072,1398]
	7	4		[ ]	[100,5,7]	[0,1072,1398]
805	1	3	[3,6,60.599]	unchanged	unchanged	unchanged
	2	3	[3,6,60.599]			
	3	3	[3,6,60.599]			
	4	1	[4,6,29.709]			
	5	1	[5,6,29.709]			
	6	1	[6,6,24.938]			
	7	1	[7,6,24.938]			
806	1	4	[3,6,60.599]	[ ]	[100,1,2]	[0,726,1139]
	2	4	[3,6,60.599]	[ ]	[100,1,100,6]	[0,726,1026,1311]
	3	4	[3,6,60.599]	[ ]	[100,1,100,6]	[0,726,1026,1311]
	4	2	[4,6,29.709]	[ ]	[100,3,100,7]	[0,734,884,1223]
	5	2	[4,6,29.709]	[ ]	[100,4,7]	[0,934,1223]
	6	2	[6,6,24.938]	[ ]	[100,5,7]	[0,1072,1398]
	7	2	[6,6,24.938]	[ ]	[100,5,7]	[0,1072,1398]

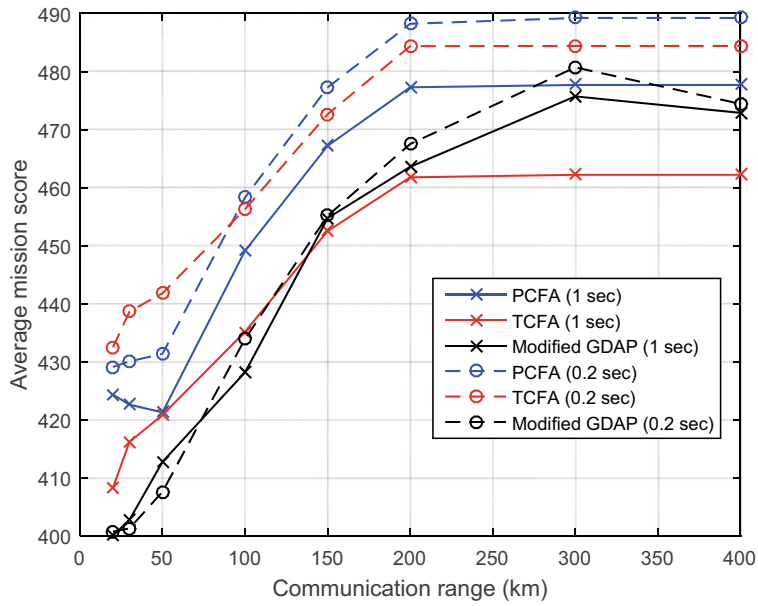


Fig. 11 Average mission score

100, 150, 200, 300, and 400 km. For each communication range, 100 random scenarios are applied. To compare the proposed algorithms with a state-of-the-art method, the GDAP algorithm [45] is modified and applied as introduced in Section 3.4.

Figure 11 presents the average mission score with respect to communication ranges. The inherent worth of tasks w_k and time-discounting factor λ are set as 100 and 0.001, respectively. For task k, its score s_k is added to the mission score if Z_k agents arrive at task k at the same time, and the second arrival by another team is not reflected in that score. The blue, red, and black solid lines indicate the results of PCFA, TCFA, and modified GDAP, respectively, when the period of TA block T_d is 1 sec. The dashed lines denote the results of the algorithms when T_d is 0.2 sec. Contrary to the expectation that TCFA shows more efficient solution than the others, PCFA performs better than the other methods for all communication ranges when T_d is 1 sec. The degradation of TCFA stems from

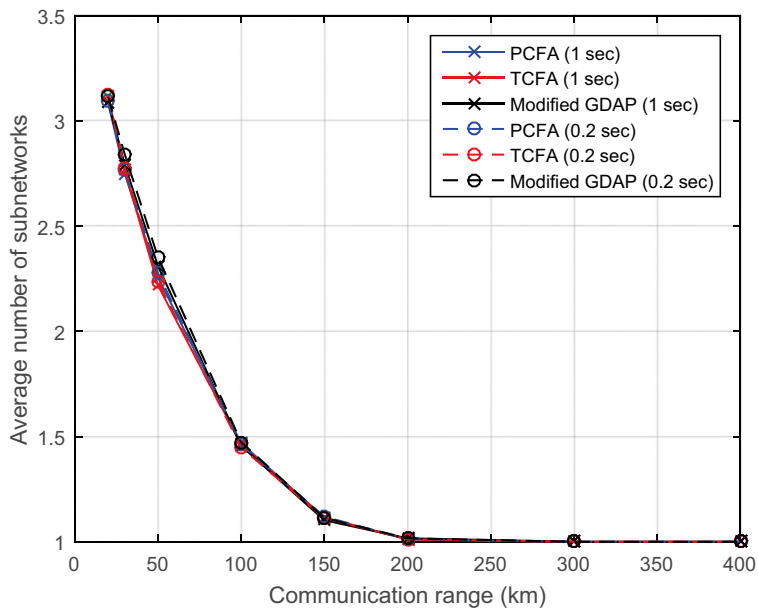


Fig. 12 Average number of subnetworks

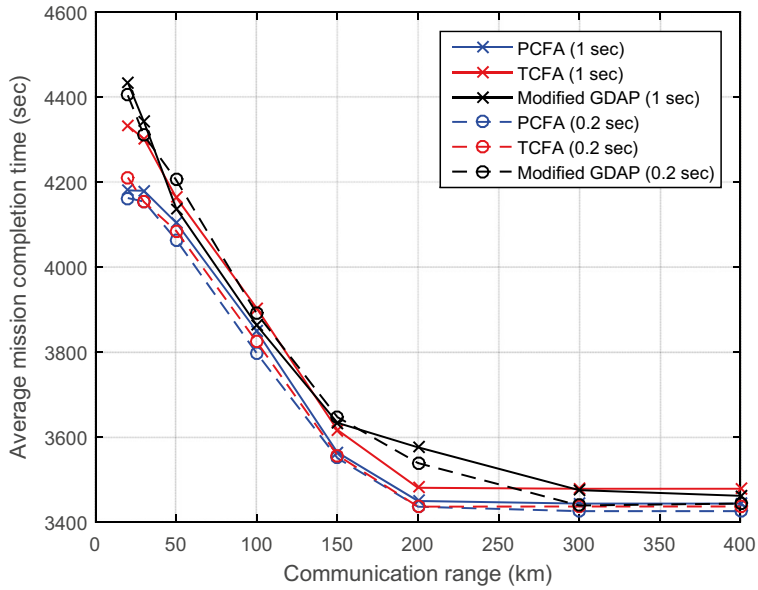


Fig. 13 Average mission completion time

the dynamic environment, which will be discussed in detail. As T_d becomes 0.2 sec, the scores of PCFA and TCFA are enhanced more distinctly than that of the modified GDAP, which implies that PCFA and TCFA are more sensitive to T_d than the modified GDAP.

As shown in Fig. 12, the network is mostly connected during the mission for the communication range beyond 200 km, and thus, TCFA solves the TA problem as in the fully connected network. Due to the time delay in phase 3, however, the performance of the TCFA is degraded. For each task, the additional time for consensus on application letters in a connected network is 9 sec because N is 10, and this delay cancels out the advantage of TCFA. Therefore, when T_d is 0.2 sec, TCFA shows better performance, and the score gap between PCFA is decreased because the additional time from phase 3 of TCFA is reduced from 9 sec to 1.8 sec. In this context, it is expected that the degradation of TCFA will be relaxed as T_d decreases.

Figure 13 shows the average of mission completion time, defined as the time spent until every agent arrives

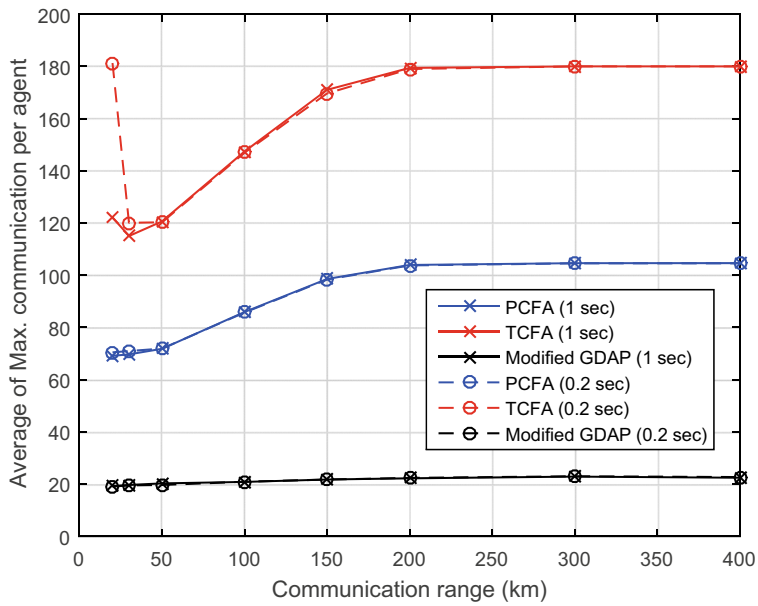


Fig. 14 Average maximum communications per agent

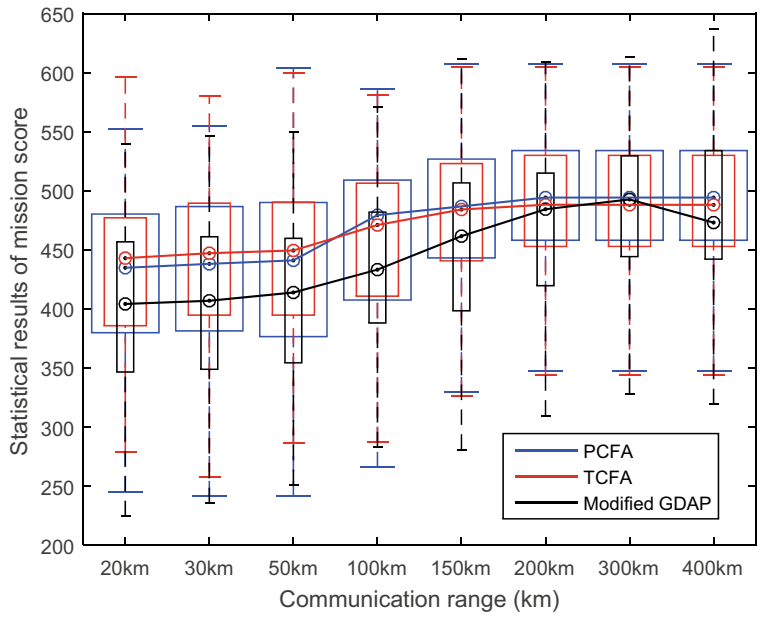


Fig. 15 Statistical results of mission score

at RP after finishing the given tasks. The average mission completion time decreases as communication range increases, which means that the dense network improves the efficiency of the TA result even when the TA algorithms are greedy. The decreasing trend of the average mission completion time is well matched with the increasing trend of the average mission score in Fig. 11.

Figure 14 exhibits the averaged number of maximum communications per agent with respect to communication ranges. Modified GDAP consumes fewer communication because it does not include the consensus process. In PCFA and TCFA, more communication are required because more resets occur over shorter communication ranges. In connected networks, the number of communication by PCFA and

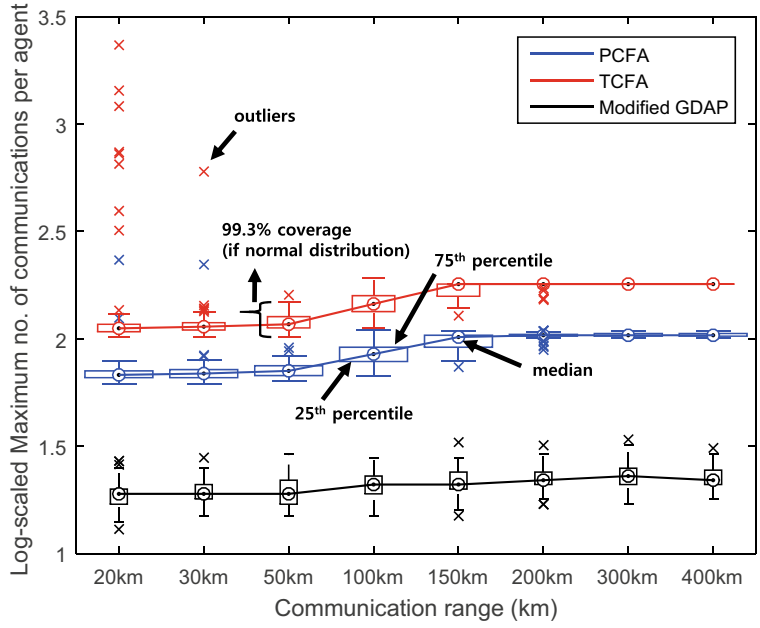


Fig. 16 Statistical results of maximum communications per agent (base=10)

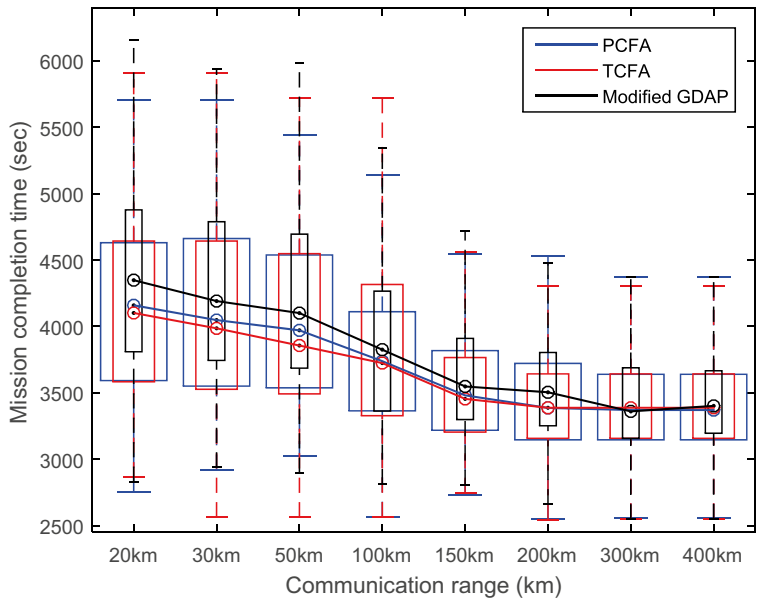


Fig. 17 Statistical results of mission completion time

TCFA are below 180, which is the upper bound from Eq. 13.

Figures 15–17 show the statistical results of Monte Carlo simulations when T_d is 0.2 sec. On each box, the edges of the box denote the 25th and 75th percentiles, the central mark is the median, and the whiskers denote 99.3 % coverage if the data are normally distributed. In Fig. 16, outliers outside the whiskers are plotted together. Considering the outliers, the proposed algorithms sometimes use significantly more communication than the modified GDAP when the communication range is short; however, the trend is relaxed for problems with longer communication ranges. Therefore, the modified GDAP can be a compromise when the communication range is much shorter than the diameter of a mission area. The proposed algorithms show better performance in terms of mission score and mission completion time by using more communication.

5.3 Discussion

Numerical simulation demonstrates that the proposed decentralized coalition formation algorithms can be applied to the dynamic environment where time-varying, isolated subnetworks may appear due to the limited communication range. Comparative study with the modified GDAP shows a trade-off relationship between communication burden and efficiency.

However, the proposed coalition formation algorithms do suffer from several limitations. First, the problem statement and the proposed algorithms neglect the constraint on finite energy of agents. By limiting the actions of advertisement and application for the case that the remaining fuel is not sufficient, the constraint can be treated intuitively. A precise model of fuel consumption, however, is hard to obtain and depends on the vehicle type such as a fixed-wing UAV or multi copter UAV. Thus, the consideration of finite energy constraint and the corresponding analysis should be conducted for future work. Second, Monte Carlo simulations are not sufficient to prove the performance based on synchronous communication. Hardware experiments and associated lessons are required to verify the proposed algorithms. Finally, the scalability analysis with respect to the computation and communication in dynamic environment was not presented. The time-varying network topology due to the limited communication range makes the convergence analysis very hard. Probability of isolated subnetworks makes this problem more challenging.

6 Conclusion

Two market-based decentralized task allocation algorithms were proposed for cooperative timing missions.

The network in dynamic environments may be disconnected during missions; thus, the proposed algorithms were extended for use in dynamic environments. With convergence proof, a scalability analysis regarding time complexity and communication load was conducted and supported by numerical results. For the comparative analysis, the state-of-the-art method is modified and implemented as a benchmark. Numerical results with extensive Monte Carlo simulations showed that the proposed algorithms perform better than the existing algorithm, but additional communications are required.

Appendix A: Directed Acyclic Graph Constraint on Dependency Graph

The TA algorithm for cooperative timing missions should provide the *proper* order of the waypoints so that the tasks can be visited simultaneously. For instance, consider a following case in which there are three agents and four tasks, $Z_1 = Z_2 = Z_3 = Z_4 = 2$, and

$$\mathbf{p}^{(1)} = [4, 3, 2],\ \mathbf{p}^{(2)} = [4, 1, 3],\ \mathbf{p}^{(3)} = [2, 1] \quad \text{(A.1)}$$

Although this path list satisfies the constraint in Eq. 2, task 2 cannot be visited simultaneously. On the other hand, when $\mathbf{p}^{(3)} = [1, 2]$, simultaneous arrivals are possible. Therefore, to generate a path list for simultaneous arrivals, the constraint on the path list should be considered.

Let us consider the dependency graph $G(\mathcal{K}, \mathcal{E}(\mathbf{P}))$ where directed edge-set $\mathcal{E}(\mathbf{P})$ is defined as follows.

$$\mathcal{E}(\mathbf{P}) = \{(\mathbf{p}^{(i)}(b), \mathbf{p}^{(i)}(b+1)) | i \in \mathcal{I}, \\ b \in \{1, ..., n(\mathbf{p}^{(i)}) - 1\}\} \quad \text{(A.2)}$$

Then, this directed graph indicates the precedence among tasks. When a directed cycle exists in the graph $G(\mathcal{K}, \mathcal{E}(\mathbf{P}))$, the path list cannot allow simultaneous arrivals, as shown in Lemma A.1.

Lemma A.1 *Let us consider the TA problem for the cooperative timing mission as defined in* Eqs. 1∼4. *When there exists a directed cycle in the dependency graph $G(\mathcal{K}, \mathcal{E}(\mathbf{P}))$, the path list $\boldsymbol{P}$ cannot allow simultaneous arrivals.*

Proof Supposing that even a directed cycle exists in the dependency graph $G(\mathcal{K}, \mathcal{E}(\mathbf{P}))$, the path list $\mathbf{P}$ can allow simultaneous arrivals.

Suppose that arbitrary three tasks among M tasks form a directed cycle in the dependency graph $G(\mathcal{K}, \mathcal{E}(\mathbf{P}))$. Let us call these tasks task k, task l, and task m in sequence, which means that some agents move in the order of $k \rightarrow l$, $l \rightarrow m$, and $m \rightarrow k$. When each task is visited simultaneously by each coalition, the arrival time for each task is uniquely determined as t_k, t_l, and t_m. Then, according to the first two sequences, $k \rightarrow l$ and $l \rightarrow m$, the arrival time will be $t_k < t_l < t_m$. However, the third sequence $m \rightarrow k$ draws $t_m < t_k$, which makes a contradiction. Without a loss of generality, it can be stated that any directed cycles in the dependency graph result in a contradiction. Therefore, the path list $\mathbf{P}$ cannot allow simultaneous arrivals when its dependency graph $G(\mathcal{K}, \mathcal{E}(\mathbf{P}))$ has any directed cycles. □

In the previous example, graph $G(\mathcal{K}, \mathcal{E}(\mathbf{P}))$ using the path of Eq. A.1 and using the modified path ($\mathbf{p}^{(3)} = [2, 1]$ is replaced by $\mathbf{p}^{(3)} = [1, 2]$) and can be plotted as Fig. 18.

In Fig. 18a, there is a directed cycle including T1, T2, and T3; thus, the path of example 1 does not satisfy Eq. 4. On the other hand, there is no directed cycle in Fig. 18b, which means that the modified path satisfies Eq. 4.

Note that the aim of Eq. 4 is similar to that of the Banker's algorithm [15], which is well-known method for deadlock avoidance. When a deadlock occurs in parallel computing environment, the execution of tasks stops until the problem becomes solved [26]. In Banker's algorithm, a virtual banker checking whether the new request may yield a deadlock or not is adopted to manage the task scheduling. While the deadlock avoidance process is run in a sequential

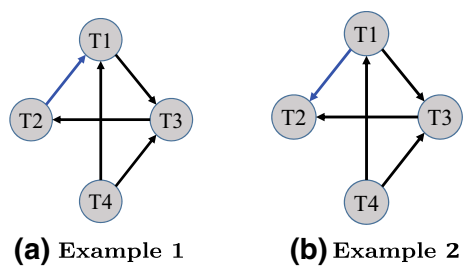


Fig. 18 Graph $G(\mathcal{K}, \mathcal{E}(\mathbf{P}))$

manner in the Banker's algorithm, batch processing method is recommended in Eq. 4.

References

1. Alighanbari, M.: Task assignment algorithms for teams of uavs in dynamic environments. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge MA (2004)
2. Alighanbari, M.: Robust and decentralized task assignment algorithms for uavs. Ph.D. dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge MA (2007)
3. Arslan, G., Marden, J.R., Shamma, J.S.: Autonomous vehicle-target assignment: A game-theoretical formulation. J. Dyn. Syst. Meas. Control. **129**(5), 584–596 (2007)
4. Balmas, F.: Displaying dependence graphs: a hierarchical approach. J. Softw. Maint. Evol. Res. Pract. **16**(3), 151–185 (2004)
5. Beard, R.W., McLain, T.W.: Multiple Uav Cooperative Search under Collision Avoidance and Limited Range Communication Constraints. In: Proceedings of the IEEE Conference on Decision and Control, pp. 25–30 (2003)
6. Bellingham, J., Tillerson, M., Richards, A., How, J.P.: Multi-Task Allocation and Path Planning for Cooperating Uavs. In: Conference on Coordination, Control and Optimization (2001)
7. Chandler, P.R.: Decentralized Control for an Autonomous Team. In: Proceedings of the AIAA 2Nd Unmanned Unlimited Conference (2003)
8. Choi, H., Kim, Y., Kim, H.: Genetic algorithm based decentralized task assignment for multiple uavs in dynamic environments. Int. J. Aeronaut. Space Sci. **12**(2), 163–174 (2011)
9. Choi, H.L., Brunet, L., How, J.P.: Consensus-based decentralized auctions for robust task allocation. IEEE Trans. Robot. **25**(4), 912–926 (2009)
10. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT University Press, Cambridge (2001)
11. Das, G.P., McGinnity, T.M., Coleman, S.A.: Simultaneous allocations of multiple tightly-coupled multi-robot tasks to coalitions of heterogeneous robots. In: Proceedings of the IEEE International Conference on Robotics and Biomimetics (2014). doi:10.1109/ROBIO.2014.7090496
12. Das, G.P., McGinnity, T.M., Coleman, S.A., Behera, L.: A distributed task allocation algorithm for a multi-robot system in healthcare facilities. J. Intell. Robot. Syst. **80**(1), 33–58 (2015)
13. Devadoss, S.L., O'Rourke, J.: Discrete and computational geometry. Princeton University Press, Princeton (2011)
14. Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: a survey and analysis. Proc. IEEE **94**(7), 1257–1270 (2006)
15. Dijkstra, E.W.: Selected writings on computing: a personal perspective. Springer, New York (1982)
16. Edison, E., Shima, T.: Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. Comput. Oper. Res. **38**(1), 340–356 (2011)
17. George, J., Sujit, P., Sousa, J.: Coalition Formation with Communication Delays and Maneuvering Targets. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference (2010)
18. Gerkey, B.P., Matari, M.J.: Sold!: Auction methods for multirobot coordination. IEEE Trans. Robot. Autom. **18**(5), 758–768 (2002)
19. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. Int. J. Robot. Res. **23**(9), 939–954 (2004)
20. Gross, J.L., Yellen, J.: Handbook of Graph Theory. CRC press, Boca Raton FL (2003)
21. Haque, M., Egerstedt, M., Rahmani, A.: Multilevel coalition formation strategy for suppression of enemy air defenses missions. J. Aerospace Inform. Syst. **10**(6), 287–296 (2013)
22. Johnson, L., Choi, H.L., How, J.P.: Hybrid Information and Plan Consensus in Distributed Task Allocation. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference (2013)
23. Karaman, S., Shima, T., Frazzoli, E.: A process algebra genetic algorithm. IEEE Trans. Evol. Comput. **16**(4), 489–503 (2012)
24. Khamis, A.M., Elmogy, A.M., Karray, F.O.: Complex task allocation in mobile surveillance systems. J. Intell. Robot. Syst. **64**(1), 33–55 (2011)
25. Manathara, J.G., Sujit, P., Beard, R.W.: Multiple uav coalitions for a search and prosecute mission. J. Intell. Robot. Syst. **62**(1), 125–158 (2011)
26. Martin, J.M.: Deadlock avoidance in distributed service oriented architectures. Master's thesis, School of Computer Science, University of Oklahoma, Norman OK (2010)
27. Maza, I., Kondak, K., Bernard, M., Ollero, A.: Multi-uav cooperation and control for load transportation and deployment. J. Intell. Robot. Syst. **57**(1-4), 417–449 (2010)
28. McLain, T.: Coordinated control of unmanned air vehicles. Technical Report ASC-99-2426 Air Vehicles Directorate of the Air Force Research Laboratory (1999)
29. Oh, G., Kim, Y., Ahn, J., Choi, H.L.: Market-Based Task Assignment for Cooperative Timing Missions over Networks with Limited Connectivity. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference (2015)
30. Ponda, S.S., Johnson, L.B., Kopeikin, A.N., Choi, H.L., How, J.P.: Distributed planning strategies to ensure network connectivity for dynamic heterogeneous teams. IEEE J. Sel. Areas Commun. **30**(5), 861–869 (2012)
31. Pujol-Gonzalez, M., Cerquides, J., Meseguer, P., Rodríguez-Aguilar, J., Tambe, M.: Engineering the decentralized coordination of uavs with limited communication range. In: Advances in Artificial Intelligence, vol. 8109, Lecture Notes in Computer Science, pp. 199–208. Springer, Berlin (2013). doi:10.1007/978-3-642-40643-0_21
32. Rawlings, J.O., Pantula, S.G., Dickey, D.A.: Applied regression analysis: a research tool. Springer, New York (1998)
33. Sandholm, T.W., Lesser, V.R.: Coalition Formation among Bounded Rational Agents. In: Proceedings of the International Joint Conference on Artificial Intelligence (1995)

34. Service, T.C., Adams, J.A.: Coalition formation for task allocation: Theory and algorithms. Auton. Agent. Multi-Agent Syst. **22**(2), 225–248 (2011)
35. Shaferman, V., Shima, T.: Unmanned aerial vehicles cooperative tracking of moving ground target in urban environments. J. Guid. Control. Dyn. **31**(5), 1360–1371 (2008)
36. Shaferman, V., Shima, T.: Task Assignment and Motion Planning for Multiple Uavs Tracking Multiple Targets in Urban Environments. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference. Chicago, IL (2009)
37. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artif. Intell. **101**(1), 165–200 (1998)
38. Shima, T., Schumacher, C.: Assigning cooperating uavs to simultaneous tasks on consecutive targets using genetic algorithms. J. Oper. Res. Soc. **60**(7), 973–982 (2009)
39. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Trans. Comput. **29**(12), 1104–1113 (1980)
40. Sujit, P., Beard, R.: Distributed Sequential Auctions for Multiple Uav Task Allocation. In: Proceedings of the IEEE American Control Conference, pp. 3955–3960 (2007)
41. Sujit, P., George, J., Beard, R.: Multiple Uav Coalition Formation. In: Proceedings of the IEEE American Control Conference, pp. 2010–2015 (2008)
42. Sujit, P., George, J., Beard, R.: Multiple Uav Task Allocation Using Particle Swarm Optimization. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference (2008)
43. Sujit, P., Manathara, J., Ghose, D., de Sousa, J.: Decentralized Multi-Uav Coalition Formation with Limited Communication Ranges. In: Handbook of Unmanned Aerial Vehicles, pp. 2021–2048. Springer, Berlin, Germany (2014)
44. Vig, L., Adams, J.A.: Coalition formation: From software agents to robots. J. Intell. Robot. Syst. **50**(1), 85–118 (2007)
45. Weerdt, M.d., Zhang, Y., Klos, T.: Multiagent task allocation in social networks. Auton. Agent. Multi-Agent Syst. **25**(1), 46–86 (2012)
46. Whitten, A.K., Choi, H.L., Johnson, L.B., How, J.P.: Decentralized Task Allocation with Coupled Constraints in Complex Missions. In: Proceedings of the IEEE American Control Conference, pp. 1642–1649 (2011)

Gyeongtaek Oh received the B.S. degree in mechanical and aerospace engineering from Seoul National University, Seoul, Korea, in 2010. He is currently pursuing the Ph.D. degree in the Department of Mechanical and Aerospace Engineering at Seoul National University. His research interests include cooperative control of multiple unmanned aerial vehicles and its real-world applications.

Youdan Kim received his B.S. and M.S. degrees in aeronautical engineering from Seoul National University, Korea, and the Ph.D. degree in aerospace engineering from Texas A&M University, in 1983, 1985, and 1990, respectively. He joined the faculty of the Seoul National University in 1992, where he is currently a Professor in the Department of Mechanical and Aerospace Engineering. His current research interests include the control system design for aircraft and spacecraft, reconfigurable flight control system, missile guidance and control.

Jaemyung Ahn received his B.S. and M.S. degrees from Seoul National University in 1997 and 1999, and Ph.D. degree in aeronautics and astronautics from MIT in 2008. He worked for the Korea Aerospace Research Institute from 1999 to 2004 and was involved in the research and development of the first liquid propellant rocket and launch vehicle of South Korea as a system engineer. From 2008 to 2010, he worked for Bain & Company as a management consultant helping strategic decisions of clients in various industrial fields. He is currently an associate professor of aerospace engineering at Korea Advanced Institute of Science and Technology (KAIST; Daejeon, South Korea). His research interests include dynamics and control of aerospace vehicles, multidisciplinary design optimization of a complex system, and design of experiments (DOE).

Han-Lim Choi is an Associate Professor of Aerospace Engineering at KAIST (Korea Advanced Institute of Science and Technology). He received his B.S. and M.S. degrees in Aerospace Engineering from KAIST, Daejeon, Korea, in 2000 and 2002, respectively, and his PhD degree in Aeronautics and Astronautics from Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2009. He then worked at MIT as a postdoctoral associate until he joined KAIST in 2010. His current research interests include planning and control of multi-agent systems, resource management in radars, and Bayesian inference for large-scale systems. He (together with Dr. Jonathan P. How) is the recipient of Automatica Applications Prize in 2011.