공학박사 학위논문

Task Allocation of Multiple UAVs for Cooperative Timing Missions

동시도달을 고려한 복수 무인기 임무할당 기법

2017 년 8 월

서울대학교 대학원

기계항공공학부

오 경 택

Task Allocation of Multiple UAVs for Cooperative Timing Missions

동시도달을 고려한 복수 무인기 임무할당 기법

지도교수 김 유 단

이 논문을 공학박사 학위논문으로 제출함

2017 년 6 월

서울대학교 대학원

기계항공공학부

오 경 택

오경택의 공학박사 학위논문을 인준함

2017 년 6 월

위 원 장 _____ (인)

부위원장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

Task Allocation of Multiple UAVs for Cooperative Timing Missions

Gyeongtaek Oh

Department of Mechanical and Aerospace Engineering

Seoul National University

APPROVED:

H. Jin Kim, Chair, Ph.D.

Youdan Kim, Vice-Chair, Ph.D.

Chan Gook Park, Ph.D.

Sungwan Kim, Ph.D.

Jaemyung Ahn, Ph.D.

Task Allocation of Multiple UAVs for Cooperative Timing Missions

A Dissertation

by

Gyeongtaek Oh

Presented to the Faculty of the Graduate School of

Seoul National University

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical and Aerospace Engineering

Seoul National University

Supervisor : Professor Youdan Kim

August 2017

Abstract

Task Allocation of Multiple UAVs for Cooperative Timing Missions

Gyeongtaek Oh
Department of Mechanical and Aerospace Engineering
The Graduate School
Seoul National University

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. As the number of agents and tasks increases, however, a greater burden is imposed on ground operators, which may cause safety issues and performance degradation accomplishing the mission. In particular, the operation requiring temporal and spatial cooperation by UAVs is significantly difficult.

This dissertation proposes autonomous task allocation algorithms for cooperative timing missions with simultaneous spatial/temporal involvement of multiple agents. After formulating the task allocation problem into integer programming problems in view of UAVs and tasks, centralized and distributed algorithms are proposed. In the centralized approach, an algorithm to find an optimal solution that minimizes the time to complete all the missions is introduced. Since the exact algorithm is time intensive, heuristic algorithms working in a greedy manner are proposed. A metaheuristic approach is also considered

to find a near-optimal solution within a feasible duration. In the distributed approach, market-based task allocation algorithms are designed. The mathematical convergence and scalability analyses show that the proposed algorithms have a polynomial time complexity. The baseline algorithms for a connected network are then extended to address time-varying network topology including isolated sub-networks due to a limited communication range. The performance of the proposed algorithms is demonstrated via Monte Carlo simulations for a scenario involving the suppression of enemy air defenses.

Keywords: Task allocation, Unmanned aerial vehicle, Cooperative timing missions, Centralized control, Distributed control

Student Number: 2010-20693

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation and Objective

Unmanned aerial vehicles (UAVs) have become widely used for civilian and military purposes because of their flying capability. The purpose of this study lies in maximizing the potential of a UAV fleet. The primary motivation of this study is the suppression of enemy air defense (SEAD) mission performed by combat UAVs as shown in Fig. 1.1.

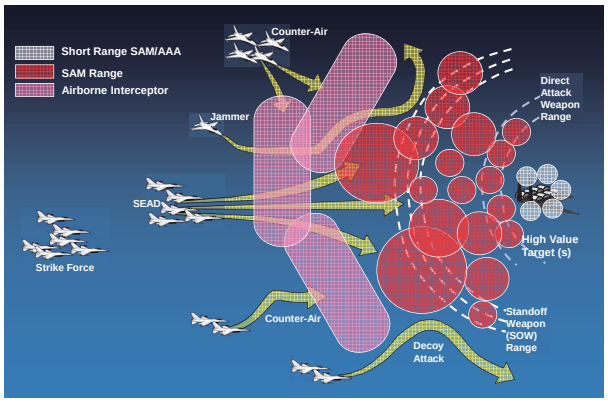


Figure 1.1 Illustrative example of SEAD mission [1]

SEAD is an offensive counter-air (OCA) mission designed to neutralize, destroy, or degrade enemy surface-based air defenses by destructive or disruptive means. [2] The fact that one-fourth of the United States (US) combat sorties

have been used for SEAD missions in recent conflicts indicates the essential role of combat aircraft in modern warfare [3]. Since SEAD aircrafts must explore and attack the enemy's air defenses with a radar system, their exposure as a target cannot be avoided, which makes the SEAD mission highly dangerous. In this context, the US department of defense (DoD) has a plan to replace manned vehicles with a fleet of UAVs for SEAD missions with the following two main attributes [4]: i) aircrew loss risk elimination, and ii) enhanced survivability through greater maneuverability beyond human tolerance.

The objective of this study is to develop an efficient task allocation (TA) algorithm that maximizes the potential of a UAV fleet. SEAD missions are considered as the primary application of this study. In particular, some task locations are required to be visited simultaneously visited by multiple UAVs to maximize the survivability and lethality of the UAVs [5]. Although much TA research has been conducted, there is insufficient research on the consideration of a task that cannot be performed by a single agent. In cooperative timing missions, agents should not only visit given waypoints, but also be on time at each waypoint. Hence, path planning and the TA problem are strongly coupled. Well-known examples of cooperative timing missions include cooperative rendezvous [6], timed attacks [5], sequential auto-landing, and coordinated ground-target suppression [7].

For more clarity, the qualitative definition of the problem considered in this paper can be stated as follows. To distinguish the problem considered in this study from other TA problems, let us call the problem considered in this study as a *target problem*. Assume that a route is a sequence of allocated tasks and the number of required UAVs to visit is predefined for each task. Now, the target problem asks "*What is the optimal set of routes for a fleet of UAVs to visit a given set of task locations?*"

1.2 Literature Survey

1.2.1 Vehicle Routing Problem

From the academic perspective, the target problem defined in Section 1.1 is a variant of the vehicle routing problem (VRP) that is an NP-hard problem in combinatorial optimization. [8] The generic definition of the *family of VRP* can be stated as follows. "*When a set of points to visit and a fleet of vehicles are given, the family of VRP determines a set of vehicle routes to visit all (or some) points with the given vehicle fleet at minimum cost; in particular, it decides which vehicle visits which points in which sequence so that all routes can be feasibly executed.*" [8]

There exist many variants of VRP, and their classification was excellently reviewed in the first chapter of Ref. [8]. The target problem can be categorized in their convention. Since each vehicle has its own starting point, the target problem is the multiple depot VRP (MDVRP). Because the vehicles are not required to return to the depot, the target problem is the open VRP (OVRP). In addition, the target problem is the VRP with multiple synchronization constraints (VRPMS) because different vehicles are needed to visit a task simultaneously. The constraints of VRPMS are called *inter-route constraints*, where the feasibility of a solution depends on how the routes and their schedules are linked. Among several synchronization constraints in inter-route constraints, the target problem has an *operation synchronization* constraint that asks different vehicles at the same or different locations to visit a task simultaneously or with precedence. Although Goel and Meisel [9] considered this kind of constraint to solve the electricity network maintenance, only centralized scheduling schemes were considered.

Table 1.1 Comparison of centralized and distributed control

	Centralized Control	Distributed Control
Pros	• Onboard computational burden is low • Optimal solution can be expected	• More robust to a failure in ground station • More adequate to a dynamic environment
Cons	• Mission area is limited around the ground station • Vulnerable to a failure in ground station	• Suboptimal performance due to local information • Information consensus between agents is needed

1.2.2 Centralized and Distributed Control

From an implementation perspective, prior research on TA can be classified into centralized and distributed approaches. In the centralized approach, a single agent or ground station receives the all agent information and sends appropriate commands to each agent. In the distributed approach, each agent makes its own decision using local information either self-obtained or through communications with neighboring agents; thus, it can address a dynamic and unexpected situation with increased agility. Additionally, this approach can use the computing power of each agent. However, consensus on situational awareness becomes more important in the distributed approach [10, 11]. The representative pros and cons are summarized in Table 1.1.

For the centralized approach, the design issue is to find a near-optimal solution within *feasible time*. For the distributed approach, computational and communicational burdens should be within an acceptable level. Additionally, dynamic network topology due to limited communication range should be properly accommodated. For both approaches, applicability to dynamic environment and convergence analysis are the main issues.

Many studies have solved a TA problem using centralized [12–17] and dis-

tributed [18–24] approaches; however, most of them considered a task that requires a single agent. Therefore, a more specific literature review of the target problem will be followed in the subsequent section. For the topic of centralized control, a review of the research conducted on the optimal coalition formation problem is made. Finally, distributed coalition formation algorithms are reviewed, focusing on limited communication range.

1.2.3 Centralized Control : Optimal Coalition Formation Problem

According to the well-known taxonomy for TA [25], the target problem is called the single-task robots (ST)–multi-robot tasks (MR) problem, where ST means that each robot is capable of executing at most one task at a time and MR indicates that each task requires multiple robots for its completion. The ST-MR problem is often referred to as a *coalition formation* problem, where a coalition is a group of agents that conduct a common task [26].

Many existing works on the coalition formation problem have been conducted by distributed artificial intelligence (DAI) researchers [27,28]. Generally, coalition formation is composed of three interacting activities [29]: i) forming a *coalition structure*, ii) solving the optimization problem of each coalition, and iii) dividing the value of the solution among coalition members, where the coalition structure is partitioned disjoint coalitions. Sandholm *et al.* [30] proved that finding an optimal coalition structure is NP-complete; therefore, various approaches have been proposed to obtain the optimal solution with reduced computational load. Rahwan *et al.* [31] proposed an anytime algorithm to find an optimal coalition structure. Chalkiadakis and Boutilier [32] provided a repeated algorithm for the problem with uncertainties between agents. The constrained coalition formation problem, where certain agents cannot be involved in the same group,

was also solved [33, 34]. Studies on bio-inspired coalition formation [35, 36] presented desirable grouping policies with theoretical and numerical analyses. The primary objective of the aforementioned approaches is to find an optimal partition between agents, and the result is a disjoint coalition structure in which each agent can join only one coalition. However, the disjoint coalition is not sufficient to describe real-world cooperative applications. For instance, if the number of tasks is greater than the number of agents and a different number of agents is required for each task, then the agents should repeatedly perform the formation, split, and re-formation procedures to satisfy the given requirements. During these procedures, each agent can improve the efficiency of the mission by being involved in multiple coalitions.

Research on the overlapping coalition formation (OCF) problem, where agents can join more than two coalitions, has been conducted for multi-sensor networks. Dang *et al.* [37] proposed an algorithm based on a branch-and-bound technique to find an optimal coalition structure that allows overlapping coalitions. Chalkiadakis *et al.* [38] described this problem as an OCF game, and the stability and balance of the solution was analyzed based on cooperative game theory. Zick *et al.* [39] analyzed the optimality and stability of the OCF game.

However, the aforementioned studies may not be appropriate for vehicle routing applications using mobile robots because the task execution order is not considered. Let us suppose that one agent is included in two different coalitions. If the agent is sensor hardware and the task is to monitor a target, then the agent can observe two targets simultaneously [37]. On the other hand, if the agent is a mobile robot and the task is to visit a target location, then the visiting order should be scheduled properly. If the visiting order is *twisted*, then one agent may not be able to visit the target together with other members because each agent has multiple appointments to meet simultaneously. Moreover, the

eventual performance of the mission is highly dependent on the task execution order.

Several researchers have studied the task execution order in the coalition formation problem. Sandholm and Lesser [29] stated that the vehicle routing problem, combined with the coalition formation problem, is too difficult to solve optimally. Shehory and Kraus [40] adopted a greedy policy, which precedes the higher valued coalition's task, to determine the task execution order. Distributed coalition formation algorithms for a multi-robot system have also been proposed [41–43]. These methods, however, focused on a distributed algorithm to make coalitions for the vehicle routing problem; thus, the performance of the methods is usually the same as that of a greedy algorithm.

Until now, the optimal coalition formation problem considering the task execution order has not been addressed sufficiently. Ramchurn *et al.* [44] augmented spatial and temporal constraints to the coalition formation problem, and simultaneity is assumed to have a synergistic effect that reduces termination time. Therefore, agents tactically cooperate to complete as many tasks as they can. Sujit *et al.* [45] proposed an optimal OCF algorithm based on particle swarm optimization (PSO) considering the task execution order. The proposed algorithm was utilized as a benchmark solution of the distributed approach [46]. However, the cost function does not systematically reflect the simultaneous arrival constraint [47].

1.2.4 Distributed Control

Among the research on distributed TA, a market-based approach has received much attention due to its computational efficiency in implementing a distributed decision-making process. After evaluating the market mechanisms for the application of multiagent coordination [48, 49], the market-based approach emerged as having good properties to describe and solve distributed coordination problems. Many variants of the earliest concept have been proposed. [22, 50, 51] Dias et al. [50] provided an excellent review and survey of many market-based coordination concepts. They defined the requirements of a market-based approach: i) a global objective function that quantifies the system designer's preferences, ii) an individual utility function which quantifies robots' preferences, and iii) a relationship between the global objective function and the individual utility function. Each participant in a virtual market makes a decision seeking its own benefit, i.e., the individual utility function. This selfish action improves the efficiency of the group, which is a global objective function. In this virtual market, resources are distributed among participants according to a market-like auction mechanism [51]. Choi et al. [22] proposed a polynomial-time algorithm, called the consensus-based bundle algorithm (CBBA), which consists of an auction-based task selection phase and a consensus phase. However, the aforementioned research did not consider a distributed coalition formation problem explicitly.

Shehory and Kraus [40] presented an iterative greedy algorithm for the distributed coalition formation that operates in exponential time. Experimental demonstrations of the distributed auction-based approach for the box pushing problem [52, 53], cooperative load transportation [54], and disaster management [55] were presented. The coalition formation for the simultaneous attack

problem was treated using a distributed scheme [46,56]. Modification of the Shehory and Kraus' algorithm and associated complexity analysis were presented for a non-overlapping coalition case [57]. A consensus-based bundle algorithm (CBBA) [22] was extended to coupled-constraint CBBA [43] to consider tightly coupled tasks. The bio-inspired coalition formation approach was proposed to apply to the suppression of enemy air defenses (SEAD) mission [36]. Das et al. proposed a market-based coalition formation that allocates multiple tasks in a centralized manner [58] as well as in a distributed manner [59]. However, the aforementioned coalition formation algorithms did 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 dealt with the problem of limited communication range. Beard and McLain [60] proposed a centralized cooperative path planning method considering distance constraints between UAVs. Sujit and Beard [61] 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 [62]. Another idea to overcome the loss of network connectivity was to make the idle agent return to the base [63]. 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. [41] proposed a variant of contract net protocol (CNP) [48] 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 distributed coalition formation algorithm for UAVs to track and destroy moving targets [64, 65] was proposed with an extensive numerical analysis of the effect of communication ranges, delays, and the problem size. However, neighboring UAVs must share their po-

sition and path information continuously to predict that their sub-network is invariant during the coalition formation process.

1.3 Research Contribution

The main contribution of this study is the design of centralized and distributed TA algorithms for cooperative timing missions with systematic problem formulation.

1.3.1 Systematic Problem Formulation

In this study, the target problem is formulated systematically. To the best of the author's knowledge, this is the first result of mathematical formulation. In the target problem, even the smallest change of one vehicle's route may yield a different mission completion time because simultaneous arrival conditions create extremely tight coupling between vehicle routes. Therefore, the objective function defined as a mission completion time is highly nonlinear and discrete with respect to vehicle routes. Although previous research [46] derived a formulation for a more generalized problem, the overlapping coalition could not be effectively handled because a constraint on the routes to resolve conflicts was not considered. This study proposed two different problem formulations that are designed as nonlinear integer programming problems. In the first formulation, the constraint on the routes is explicitly based on graph theory. In the second formulation, the optimization variable that inherently satisfies the graph-theoretic constraint is adopted.

1.3.2 Design of a Centralized TA Algorithm for a Cooperative Timing Mission

In this study, various algorithms are designed to solve the target problem in a centralized manner. To obtain an exact optimal solution, an efficient exhaustive search method is proposed. As the exact algorithm requires significant computational time, two heuristic algorithms are proposed: one has a polynomial time complexity and the other has an exponential time complexity. To find a proper trade-off between performance and computation time, PSO is applied to both problems. The performances of the proposed algorithms are analyzed and compared through numerical simulation.

1.3.3 Design of a Distributed TA Algorithm for a Cooperative Timing Mission

Two market-based distributed TA algorithms are proposed for a dynamic environment with a limited communication range. In the proposed algorithms, a leader of a coalition is elected by other agents. Each agent's position and plan do not have to be continuously synchronized. Analyses on convergence and scalability are performed that are also supported by numerical results. The performances of the proposed algorithms are demonstrated by Monte Carlo simulations.

1.4 Dissertation Organization

This dissertation is organized as follows:

Chapter 1 introduces subjects related to this study. The motivation and objective of this study are described in Section 1.1, and related research works are provided in Section 1.2. The contribution and outline of this study are stated in Section 1.3 and Section 1.4.

Chapter 2 provides two formulations of the target problem.

Chapter 3 presents centralized task allocation algorithms. Section 3.2 describe an exact algorithm for the target problem. Section 3.3 and 3.4 describes heuristic algorithms which approximate the exact algorithm. In Section 3.5 and 3.6, two particle swarm optimization schemes are provided. Simulation results are provided in Section 3.7.

Chapter 4 presents distributed task allocation algorithms. The proposed schemes are provided in Section 4.2 and 4.3, respectively. Section 4.4 describes a benchmarking algorithm for comparative studies. Properties of the proposed algorithms are summarized in Section 4.5. In Section 4.6, the proposed algorithms are extended deal with a dynamic problem. Numerical results are shown in Section 4.7.

Finally, Chapter 5 provides concluding remarks and future research works.

Chapter 2

Problem Statement

Starting with an introduction of underlying assumptions, this chapter provides formulations of task allocation problem for cooperative timing missions. First, the problem is formulated as an integer programming problem in view of agents. Second, a binary integer programming problem is presented in view of tasks. Finally, more simplified form of task-based problem is proposed.

2.1 Assumptions

Throughout this dissertation, the following assumptions are considered.

Assumption 2-1. The aim of TA algorithm is to produce a task visiting order and corresponding arrival time for each agent.

Assumption 2-2. Every agent moves in two-dimensional space with its own constant speed. Dynamics of agents such as maximum turn radius are neglected.

Assumption 2-3. Each agent has a finite number of homogeneous resources and uses one resource at one task.

Assumption 2-4. The number of required agents for each task is predefined.

Assumption 2-5. There exists a feasible allocation of the given problem. In other words, given tasks can be accomplished by given agents within finite time.

Assumption 2-6. Agents can control their arrival time by loitering around the safe boundary of the task. It means that agents wait for their coalitions until all

members arrive at the specified boundary of the given task. The execution time of a task is then decided by the arrival time of the latest member. Therefore, once a feasible visiting order is given for all agents, corresponding arrival times can be determined as well.

Assumption 2-7. Collisions between UAVs are autonomously avoided. Delayed time due to collision avoidance can be resolved by a momentary speed control.

2.2 Agent-based Formulation

Let us consider a TA problem with N agents and M tasks, where some task locations should be simultaneously visited by multiple agents, as shown in Fig. 2.1. The number inside the parenthesis of each task location indicates the number of required agents for each task, which is *predefined* based on the properties of the task. In this setting, the aim of this study is to determine all agents' task visiting orders that maximize the objective function. This problem can be formulated in terms of the integer programming problem as follows,

$$\underset{\mathbf{P}}{\text{Maximize}} \;\; J = s(t_1(\mathbf{P}), t_2(\mathbf{P}), ..., t_M(\mathbf{P})) \tag{2.1}$$

$$\text{subject to} \;\; \texttt{card}(\mathbf{a}_k) = z_k, \;\; \forall k \in \mathcal{K} \tag{2.2}$$

$$\texttt{card}(\mathbf{p}^{(i)}) \leq y^{(i)}, \quad \forall i \in \mathcal{I} \tag{2.3}$$

$$\texttt{isDAG}(G) = 1, \quad G = (\mathcal{K}, \mathcal{E}(\mathbf{P})) \tag{2.4}$$

where $\mathcal{I} \triangleq \{1, 2, ..., N\}$ and $\mathcal{K} \triangleq \{1, 2, ..., M\}$ are the sets of indices of the agents and tasks, respectively, and $\mathbf{P} = (p_{i,m}) \in (\{0\} \cup \mathcal{K})^{N \times M}$ is a $N \times M$ path matrix having the information regarding all agents' task visiting orders. For instance, $p_{i,m} = k$ means that task k is the m-th task of agent i, and $p_{i,m} = 0$ denotes that no task is allocated to agent i as the m-th order. The path vector of agent i, $\mathbf{p}^{(i)}$, is the i-th row of $\mathbf{P}$. The objective function J can be defined as the score function $s(\mathbf{P})$ in Eq. (2.1), where t_k is the termination time of the task k. Equation (2.2) defines a constraint on the size of the coalition for task k, where $\mathbf{a}_k$ is a coalition vector containing indices of agents assigned to the task k, $\texttt{card}(\cdot)$ denotes the number of nonzero elements in a set (or a vector), and z_k ($\leq N$) is the number of agents required for the task k. A $N \times M$ coalition matrix $\mathbf{A} = (a_{i,k}) \in \{0, 1\}^{N \times M}$ contains the information regarding

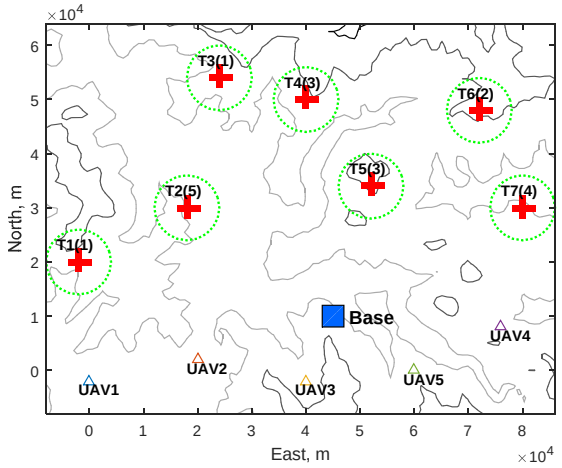


Figure 2.1 Example scenario of cooperative timing mission

the overlapping coalition structure, where $a_{i,k} = 1$ if agent i is involved as a coalition member for task k and 0 otherwise. Because $\mathbf{A}$ is uniquely determined by $\mathbf{P}$, $\mathbf{a}_k$ can also be specified when $\mathbf{P}$ is given. An illustrative example is introduced in Fig. 2.2 for the aforementioned notations. Equation (2.3) restricts the maximum number of allowable tasks for agent i to $y^{(i)}$ ($\leq M$). Additionally, Eq. (2.4) is introduced to disregard the TA results when the involved agents fail to simultaneously arrive at the allocated task locations. This process can be conducted by filtering out the TA results that generate the directed cycle in

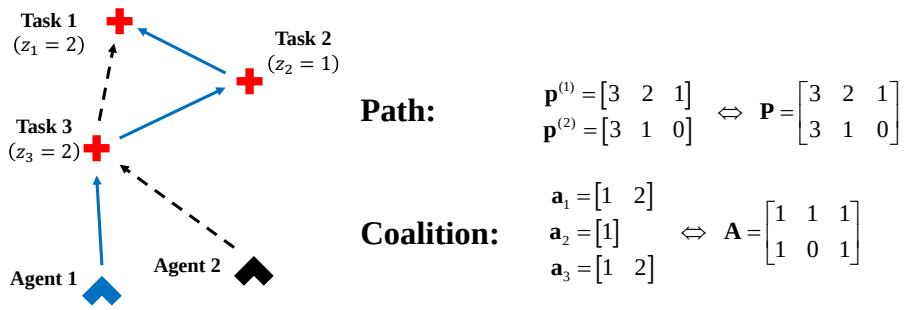


Figure 2.2 Illustrative example for the notations in problem statement

the dependency graph $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$, which represents the precedence between the allocated tasks [66]. The directed edge set $\mathcal{E}(\mathbf{P})$ is defined as follows,

$$\mathcal{E}(\mathbf{P}) = \{(p_{i,j}, p_{i,j+1}) | i \in \mathcal{I},\ j \in \{1, ..., \texttt{card}(\mathbf{p}^{(i)}) - 1\}\} \tag{2.5}$$

The simultaneous arrival fails when the dependency graph contains a directed cycle. Therefore, designating the type of the dependency graph as a directed acyclic graph (DAG), which is a directed graph with no directed cycles [67], results in the filtering. The function $\texttt{isDAG}(G)$ is one if the graph G is DAG and zero otherwise. A detailed explanation regarding the DAG constraint is provided in the Appendix.

In this study, the objective function J is represented by the termination times of given tasks. Because tasks should be visited simultaneously, t_k is defined as the required time of the latest agent among the coalition for the task k, which is expressed as

$$t_k(\mathbf{P}) = \underset{i \in \mathbf{a}_k}{\texttt{max}}\ \ (t_w(i,k) + t_{ETA}(i,k)) \tag{2.6}$$

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

$$t_{ETA}(i,k) = \begin{cases} \left\| \mathbf{x}_k - \mathbf{x}^{(i)} \right\| / v^{(i)}, & \text{if } \ p_{i,1} = k \\ t_j + \| \mathbf{x}_k - \mathbf{x}_j \| / v^{(i)}, & \text{otherwise} \end{cases} \tag{2.7}$$

where $\mathbf{x}_k$ is the position vector of task k, $\mathbf{x}^{(i)}$ is the initial position vector of the agent i, and $v^{(i)}$ is the average speed of the agent i. When the task k is not the first task of agent i, the task j denotes the task conducted by the agent i

prior to the task k. Note that j can be expressed as $j = p_{i,m}$, where the index m satisfies $p_{i,m+1} = k$. In this study, the objective function is defined as follows to minimize the total mission completion time t_c.

$$J = -t_c = -\max_{k \in \mathcal{K}} t_k \tag{2.8}$$

As mentioned in Section 1.2, the target problem defined in Eqs. (2.1)–(2.4) is a variant VRP, which is NP-hard [8]. Because the original VRP does not consider multiple depots and multiple synchronization constraints, the problem considered in this study is at least as complex as the VRP.

2.3 Task-based Formulation

In the previous section, the decision variable is set as task visiting orders of each agent, which is intuitive and conventional [46, 68]. The DAG constraint in Eq. (2.4), however, is not a typical form of constraint for integer programming problem. This section introduces an alternative formulation to eliminate the DAG constraint.

Based on the fact that a dependency graph can be topologically sorted if and only if the graph is DAG, the DAG constraint can be eliminated by setting the precedence order of each task as an additional decision variable. The task visiting orders of each agent $\mathbf{P}$ are also uniquely determined when the coalition matrix $\mathbf{A} = (a_{i,k}) \in \{0,1\}^{N \times M}$ and the precedence order matrix $\mathbf{V} = (v_{k,m}) \in \{0,1\}^{M \times M}$ are given. Therefore, the original TA problem in Eqs. (2.1)–(2.4) can be reformulated as follows,

$$\underset{\mathbf{A},\mathbf{V}}{\text{Maximize}} \quad J = s(t_1(\mathbf{P}), t_2(\mathbf{P}), ..., t_M(\mathbf{P})) \tag{2.9}$$

$$\text{subject to} \quad \sum_{i=1}^{N} a_{i,k} = z_k, \quad \forall k \in \mathcal{K} \tag{2.10}$$

$$\sum_{k=1}^{M} a_{i,k} \leq y^{(i)}, \quad \forall i \in \mathcal{I} \tag{2.11}$$

$$\sum_{m=1}^{M} v_{k,m} = 1, \quad \forall k \in \mathcal{K} \tag{2.12}$$

$$\sum_{k=1}^{M} v_{k,m} = 1, \quad \forall m \in \mathcal{K} \tag{2.13}$$

where $a_{i,k} = 1$ if agent i is involved as a coalition member for task k, and 0 otherwise. The precedence order $v_{k,m} = 1$ if task k has m-th priority, and 0 otherwise. For instance, the path matrix in Fig. 2.2 can be represented by the

following matrices. Note that all the decision variables are binary and the DAG constraint is not required in this setting.

$$\mathbf{P} = \begin{bmatrix} 3 & 2 & 1 \\ 3 & 1 & 0 \end{bmatrix} \Leftrightarrow \mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \ \mathbf{V} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{2.14}$$

2.4 Simplified Form of Task-based Formulation

The aforementioned task-based formulation succeeded in removing the DAG constraint. However, the number of constraints become four, which is larger than that of the agent-based formulation. Also, the dimension of the decision variables increases from (NM) to $(NM + M^2)$. To reduce the number of constraints and dimensions of the task-based formulation, the coalition matrix $\mathbf{A}$ and the precedence order matrix $\mathbf{V}$ are replaced by the coalition vector $\mathbf{a}_k$ and the precedence order vector $\mathbf{v}\in \mathcal{K}$. For instance, the precedence order matrix $\mathbf{V}$ in Eq. (2.14) can be represented as $\mathbf{v} = [3\ 2\ 1]$, which means that the precedence order is $3 \rightarrow 2 \rightarrow 1$. Then, the Eqs. (2.12)–(2.13) are satisfied inherently. By designating the dimension of $\mathbf{a}_k$ to z_k, Eq. (2.10) can be satisfied always. Therefore, the simplified form of task-based TA problem in Eqs. (2.9)–(2.13) can be reformulated as follows,

$$\underset{\mathbf{a}_k,\mathbf{v}}{\text{Maximize}}\ \ J = s(t_1(\mathbf{P}), t_2(\mathbf{P}), ..., t_M(\mathbf{P})) \tag{2.15}$$

$$\text{subject to}\ \ \texttt{card}(\mathbf{p}^{(i)}) \leq y^{(i)},\ \ \forall k \in \mathcal{K},\ \forall i \in \mathcal{I} \tag{2.16}$$

In summary, the problem can be completely specified by the vectors and matrices in Table. 2.1. The solution of the problem is i) the path matrix $\mathbf{P}$, or ii) the pair of the coalition matrix $\mathbf{A}$ and the precedence order matrix $\mathbf{V}$, or iii) the pair of the coalition vectors $\mathbf{a}_k$ and the precedence order vector $\mathbf{v}$. In subsequent chapters, the solution of the target problem formulated in three different forms will be discussed in the centralized and distributed ways. After introducing centralized schemes in Chap. 3, distributed schemes will be followed in Chap. 4.

Table 2.1 Summary of variables

Group	Symbol	Description (input*, output†, internal variables§)
Agent i	$\mathbf{x}^{(i)}$	Position vector *
	$v^{(i)}$	Average speed *
	$y^{(i)}$	Number of resources *
	$t_w(i,k)$	Working time for task k *
	$\mathbf{P}$	Path matrix †
	$\mathbf{p}^{(i)}$	Path vector (i-th row of $\mathbf{P}$) §
	$p_{i,m}$	m-th visit task (m-th element of $\mathbf{p}^{(i)}$) §
	$t_{ETA}(i,k)$	Estimated time of arrival to task k §
Task k	$\mathbf{x}_k$	Position vector *
	z_k	Number of required agents *
	$\mathbf{A}$	Coalition matrix †
	$\mathbf{a}_k$	Coalition vector (index set of assigned agents) †
	$a_{i,k}$	(i,k)-element of $\mathbf{A}$ §
	$\mathbf{V}$	Precedence order matrix †
	$\mathbf{v}$	Precedence order vector †
	$v_{k,m}$	(k,m)-element of $\mathbf{V}$ §
	t_k	Termination time †

Chapter 3

Centralized Task Allocation

This chapter provides centralized algorithms to solve the TA problem. In the centralized TA, the master control center or leader agent solves the problem using exact information of the whole mission, and then transmits the TA results to the fleet. In this chapter, first, the enumerative method is proposed to obtain the exact optimal solution. Second, two greedy algorithms are proposed as a heuristic approach. Finally, two PSO algorithms are presented as a metaheuristic approach. To demonstrate and compare the performance of the proposed algorithms, numerical simulations for a SEAD mission are conducted.

3.1 Assumptions

Throughout this chapter, the following assumptions are considered.

Assumption 3-1. The mission control center can achieve precise information on the entire UAVs and tasks.

Assumption 3-2. The mission control center can broadcast the commands to the entire UAVs.

3.2 Exact Algorithm

In this section, the exact algorithm to solve the aforementioned TA problem is considered. The exact algorithm considers all possible combinations of the solutions to find the optimal solution, which guarantees finding the optimal solution of the integer programming problem. The exact algorithm might be unrealistic due to its computational load. The optimization variable in Eq. (2.1) is the path matrix $\mathbf{P} \in (\{0\} \cup \mathcal{K})^{N \times M}$, and the number of possible solutions can be roughly estimated as $(M+1)^{NM}$. For example, it is $5^{16} \approx 1.5 \times 10^{11}$ for a case of $N = M = 4$, which is computationally infeasible. Moreover, constraint equations, Eqs. (2.2)–(2.4) should be checked for each candidate. Therefore, the full factorial experiment may be intractable with respect to the computation time.

In this study, the main idea is that the enormous search space can be fairly reduced by pruning the infeasible spaces that do not satisfy the constraint equations of Eqs. (2.2)–(2.4). Using the current optimization variable $\mathbf{P}$, however, the pruning is hard to be implemented. Therefore, the simplified form of task-based formulation introduced in Section 2.4 is considered for implementing the exact algorithm. Possible coalitions for $\mathbf{a}_k$ and the precedence task orders $\mathbf{v}$ can be obtained by utilizing combinations and permutations. That is, the number of possible solutions N_T can be calculated as follows,

$$N_T = \left(\prod_{k=1}^{M} \begin{pmatrix} N \\ z_k \end{pmatrix} \right) \times M! \tag{3.1}$$

For the aforementioned instance, where $N = M = 4$ and all $z_k = 2$, N_T is $6^4 \approx 3 \times 10^4$, which is far less than 1.5×10^{11}. Moreover, the simplified form of the task-based formulation has only one constraint, which means that

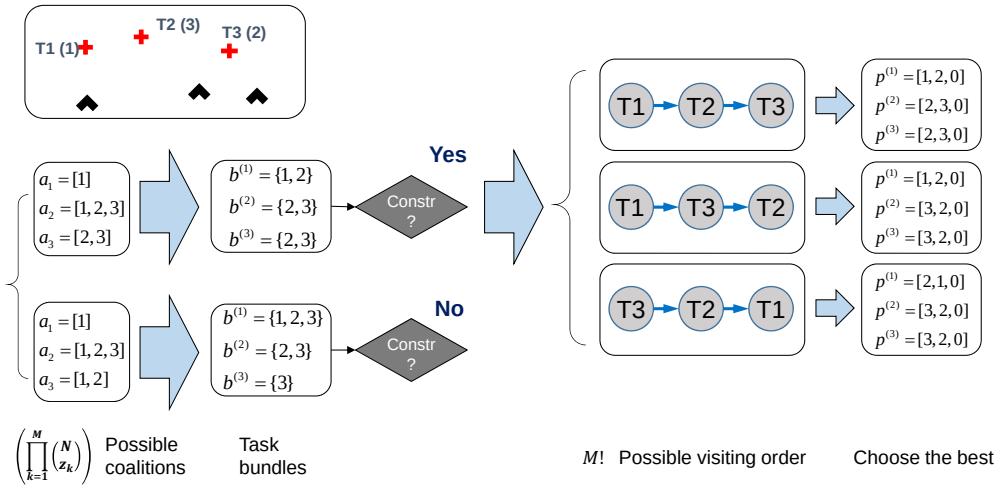


Figure 3.1 Illustrative example of the exact algorithm

substantial part of infeasible solution space is pruned successfully.

The detailed procedures of the exact algorithm are as follows. First of all, all possible coalitions are calculated to decide *which agents are allocated for each task*. For each coalition candidate, unordered task lists for each agent is built, which is called a task bundle. When the task bundle satisfies the constraint in Eq. (2.16), path vectors for each agent are given according to $M!$ possible candidates having the corresponding visiting order. Finally, the exact algorithm selects the best path vector as an optimal solution. The overall procedure is shown in Fig. 3.1 with an illustrative example.

3.3 Agent-based Sequential Greedy Algorithm : A-SGA

In this section, a traditional sequential greedy algorithm (SGA) [47] is briefly summarized. The SGA matches a capable coalition with its corresponding task repeatedly in a short-term perspective. That is, the SGA decides a pair of the coalition leader and the target task by choosing the match that has the shortest ETA. After the coalition leader and the corresponding task are determined, agents having less ETA for the target task are selected as coalition members. This procedure is repeated until the whole tasks are allocated. To distinguish between the traditional SGA and the proposed SGA, which will be introduced in the following section, the traditional one is called agent-based SGA (A-SGA) in this study.

The merit of the A-SGA is that the DAG constraint is automatically satisfied, because newly allocated task is augmented at the end of the current sequence of the tasks, thereby a directed cycle does not appear. Also, computational burden can be significantly reduced compared to the exact algorithm. Therefore, the A-SGA may be a possible choice for practical application of the cooperative timing mission. The detailed procedure of the A-SGA is summarized in Algorithm 3.1.

Algorithm 3.1 Agent-based Sequential Greedy Algorithm (A-SGA)

procedure $\mathbf{P}$=A-SGA($\mathcal{I}, \mathcal{K}$)
  $\mathbf{P} = \mathbf{0}_{N \times M}$
  $\mathcal{K}_0 = \mathcal{K}$
  for iter=1:$\texttt{card}(\mathcal{K})$ **do**
    $(i^*, k^*) = \underset{(i,k) \in \mathcal{I} \times \mathcal{K}_0}{\texttt{arg min}} \; (t_w(i,k) + t_{ETA}(i,k))$
    $b^* = \texttt{card}(\mathbf{p}^{(i^*)}) + 1$
    $p_{i^*,b^*} = k^*$
    $\mathcal{I}_0 = \mathcal{I} \setminus \{i^*\}$
    for z=1:$(z_{k^*} - 1)$ **do**
      $j^* = \underset{j \in \mathcal{I}_0}{\texttt{arg min}} \; (t_w(j,k^*) + t_{ETA}(j,k^*))$
      $c^* = \texttt{card}(\mathbf{p}^{(j^*)}) + 1$
      $p_{j^*,c^*} = k^*$
      $\mathcal{I}_0 = \mathcal{I}_0 \setminus \{j^*\}$
    end for
    $\mathcal{K}_0 = \mathcal{K}_0 \setminus \{k^*\}$
  end for
end procedure

3.4 Task-based Sequential Greedy Algorithm : T-SGA

In A-SGA, two greedy decisions are made for each matching between a task and its coalition: i) a target task and its corresponding coalition leader are determined concurrently by greedy sense, and then ii) several agents are selected as the coalition members by greedy sense. In cooperative timing missions, however, the first decision is much more important than the second one because task execution order is fixed by the first one. In addition, greedy selection of team members is a reasonable strategy because termination time of a task is determined by a latest member. Based on these intuitions, the task-based sequential greedy algorithm (T-SGA) is proposed in this study.

The fundamental idea of the T-SGA is to replace the important decision of the A-SGA with the process of exact algorithm, which means that all possible task execution orders are investigated. For instance, $M!$ task execution orders are considered when M tasks are given. Coalition members including a leader for each task are determined in a greedy manner. Among the resultant objective functions for each task execution order, the case having maximum-value is selected as the solution of the T-SGA. It can be seen that T-SGA solves the task-based problem by using two-step approach because greedily pruned space of $\mathbf{A}$ is investigated for each possible $\mathbf{V}$, which is the reason why it is called task-based SGA. Overall procedure of the T-SGA is shown in Fig. 3.2.

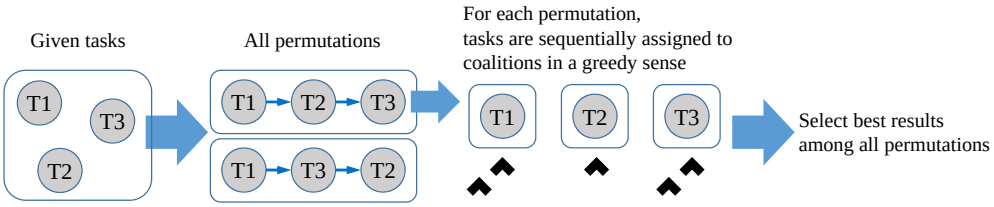


Figure 3.2 Overall procedure of T-SGA

The solution of the T-SGA naturally satisfies Eq. (2.2). It is known that a graph can be sorted topologically if and only if the graph is DAG [69]. The topological sorting of a directed graph G is a linear ordering of all the vertices; if G contains an edge (u, v), then u appears before v in the order. The illustrative example of the topological sorting of dependency graph G is shown in Fig. 3.3. Because the T-SGA considers all possible permutations of task execution order, all feasible domain of Eq. (2.2) can be investigated by the T-SGA.

Note that the T-SGA is not a polynomial algorithm; it may require much computation time for big problems with large (N, M). For problems with moderate size of (N, M), however, the computation time will not become a serious issue because the computational load for t_{ETA} is not heavy. In addition, the solution of the T-SGA is always better than or at least same as that of the A-SGA, because candidates of the task execution order of the T-SGA include the task execution order determined by the A-SGA.

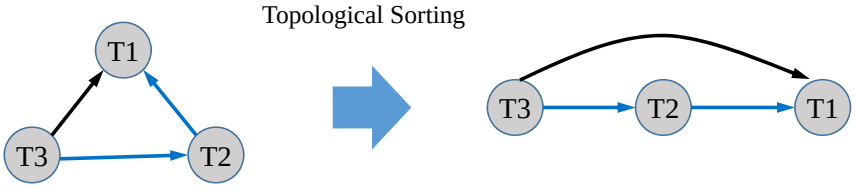


Figure 3.3 Example of topological sorting

3.5 Agent-based Particle Swarm Optimization : A-PSO

In the previous section, the exact algorithm was presented to find an exact optimal solution of the TA problem. However, it is obvious that significant time is required as the size of the problem increases. In this section, PSO, a metaheuristic approach, is adopted to solve the problem with reduced computational load. The original idea to solve the problem by using PSO was proposed by Sujit *et al.* [45]. In this study, the existing PSO method is revised to solve the problem in a systematic manner.

3.5.1 Preliminaries on PSO

Let us briefly summarize the PSO algorithm. The PSO, a population-based stochastic optimization technique developed by Kennedy and Eberhart [70], encodes the optimization variable as a position vector of a particle, which is updated by a velocity vector of the particle. Recent and comprehensive review on PSO can be found in [71, 72].

Assuming that the dimension of the optimization variable is N_d, the update rule of the PSO can be represented as follows,

$$V_s^{n+1} = \omega\, V_s^n + c_1 r_{s1}^n (P_s^n - X_s^n) + c_2 r_{s2}^n (P_g^n - X_s^n) \tag{3.2}$$

$$X_s^{n+1} = \texttt{round}(X_s^n + \chi\, V_s^{n+1}) \tag{3.3}$$

where X_s^n and V_s^n are the N_d–dimensional position vector and the velocity vector of the s-th particle in the n-th iteration ($s \in \{1, 2, ..., N_s\}$, $n \in \{1, 2, ..., N_i\}$), respectively, N_s is the number of particles, N_i is the number of iterations, P_s^n is the s-th particle's best position, P_g^n is the swarm's best position, ω is the

inertia weight, positive scalars c_1 and c_2 are the cognitive and social parameters, respectively, r_{s1}^n and r_{s2}^n are N_d–dimensional row vectors whose elements are uniformly distributed random variables within $[0, 1]$, and χ is a constriction factor. Note that the tuning parameters of the PSO are ω, c_1, c_2, and χ. The physical meaning of ω is the level of belief in the previous decision (velocity). Large ω leads to global exploration, whereas small ω focuses on local exploration nearby the best positions (P_s^n, P_g^n). Therefore, gradually declining ω is recommended in general. Parameters c_1 and c_2 compensate for the differences of X_s^n to P_s^n and X_s^n to P_g^n, respectively. In most implementations of the PSO, χ is calculated as follows [73–75],

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{3.4}$$

where $\phi = c_1 + c_2 > 4$. Typically, ϕ is set to 4.1, with $c_1 = c_2 = 2.05$ [73]. In this study, the type of optimization variable is an integer, and therefore, the `round` operation is adopted to round its argument to the nearest integer [74]. Figure 3.4 shows the procedure of the standard PSO algorithm.

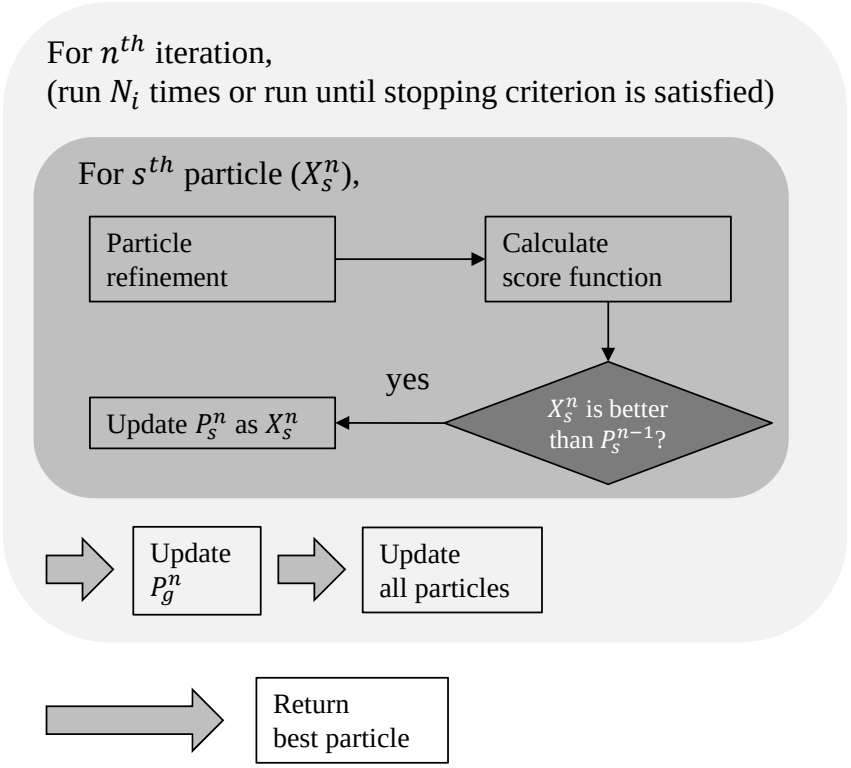


Figure 3.4 Overall procedure of PSO

3.5.2 Particle Encoding

To solve the TA problem defined by Eqs. (2.1)–(2.4) using PSO, the optimization variable should be represented as the position vector of a particle, which can be performed by reshaping the N by M path matrix $\mathbf{P}$ into a NM–dimensional column vector X_s^n [45]. Because the path matrix is encoded as a particle, this approach is called agent-based PSO (A-PSO). For example scenario of $(N, M) = (2, 3)$, the path matrix can be transformed into the position vector of a particle as follows,

$$\mathbf{P} = \begin{bmatrix} 1 & 3 & 0 \\ 2 & 3 & 0 \end{bmatrix} \Leftrightarrow X_s^n = \begin{bmatrix} 1 & 3 & 0 & 2 & 3 & 0 \end{bmatrix}^{\mathrm{T}} \tag{3.5}$$

3.5.3 Particle Refinement

A particle may have inappropriate elements that prevent evaluation of the score function. On the other hand, in the previous study [45], this issue was not addressed explicitly. In this study, three refinement schemes are proposed to treat this problem. First, if an element of the position vector is outside the range $[0, M]$, then the element is replaced by a random integer in $[0, M]$. Second, if a row vector of the path matrix contains a zero between the nonzero elements, then the zero is moved to the right, as follows,

$$X_s^n = \begin{bmatrix} 1 & 0 & 3 & 2 & 3 & 0 \end{bmatrix}^{\mathrm{T}} \Rightarrow \mathbf{P} = \begin{bmatrix} 1 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix} \Rightarrow \mathbf{P} = \begin{bmatrix} 1 & 3 & 0 \\ 2 & 3 & 0 \end{bmatrix} \tag{3.6}$$

Finally, if a row vector of the path matrix has more than two identical tasks, then the row vector is replaced by a random permutation of N_x elements chosen from one to M, where N_x is the number of nonzero elements of the original row

vector. For instance,

$$X_s^n = \begin{bmatrix} 1 & 0 & 1 & 2 & 3 & 0 \end{bmatrix}^{\mathrm{T}} \Rightarrow \mathbf{P} = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 3 & 0 \end{bmatrix} \Rightarrow \mathbf{P} = \begin{bmatrix} 3 & 1 & 0 \\ 2 & 3 & 0 \end{bmatrix} \tag{3.7}$$

Because the update law of the PSO algorithm does not consider the feasible space of the position vector, these refinements may reduce the computation time by recovering the meaningless particles into the feasible particles that satisfy the problem constraints.

3.5.4 Score Calculation Considering DAG Constraint

The existing algorithm [45, 46] has two limitations in the score calculation process for each particle. First, the generated path from the particle does not guarantee the simultaneous arrivals to the tasks because the DAG constraint in Eq. (2.4) is not considered. In this study, to deal with this issue, a check logic of the DAG constraint is included. Second, the calculation order of the termination time for each task, i.e., t_k, is not systematic. The topological sorting scheme is considered in this study to treat this problem.

Let us consider following path vectors which satisfy the DAG constraint.

$$\mathbf{p}^{(1)} = [4\ 3\ 2\ 0],\ \mathbf{p}^{(2)} = [4\ 1\ 3\ 0],\ \mathbf{p}^{(3)} = [1\ 2\ 0\ 0] \tag{3.8}$$

The *valid* calculation order of the termination time can be determined by using the modified path matrix $\overline{\mathbf{P}}$, which is obtained by shifting elements in each row to the right/left to make each column have only one task, i.e., the same number

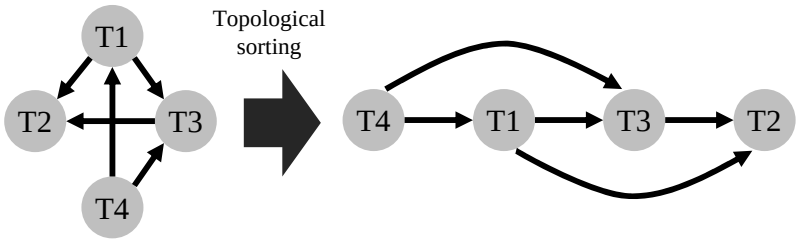


Figure 3.5 Illustrative example of topological sorting

in each column, as follows,

$$\mathbf{P} = \begin{bmatrix} 4 & 3 & 2 & 0 \\ 4 & 1 & 3 & 0 \\ 1 & 2 & 0 & 0 \end{bmatrix} \Rightarrow \overline{\mathbf{P}} = \begin{bmatrix} 4 & 0 & 3 & 2 \\ 4 & 1 & 3 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix} \tag{3.9}$$

In this manner, the valid visiting order for each task is revealed as $4 \rightarrow 1 \rightarrow 3 \rightarrow 2$, and the calculation order of the termination time is t_4, t_1, t_3, and t_2. Note that for some path vectors, $\overline{\mathbf{P}}$ cannot be obtained. Conditions on the path vectors with proper visiting schedule is summarized in Appendix. The above procedure can be carried out by sorting the graph $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$ topologically. The topological sorting of a directed graph G is a linear ordering of all the vertices; if G contains an edge (u, v), then u appears before v in the order [69]. According to the definition of $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$, the vertices denote tasks to be performed, while the edges are precedents between two tasks. Therefore, topological sorting of $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$ yields a valid sequence for the tasks to be performed, which is equivalent to the valid calculation order of the termination time. Moreover, a graph can be topologically sorted if and only if the graph is DAG, which means that $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$ satisfying Eq. (2.4) can be topologically sorted, as shown in Fig. 3.5.

Once the calculation order is determined, termination time for each task

Algorithm 3.2 Score of a Particle

procedure J=SCORE($\mathbf{P}$)
  if $\mathbf{P}$ satisfies Eqs. (2.2)–(2.4) **then**
    $\mathbf{k}$ = topologically sorted vertices of $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$
    Initialize t_k ($\forall k \in \mathcal{K}$) as 0
    for $idx = 1 : \mathtt{card}(\mathcal{K})$ **do**
      $k = \mathbf{k}(idx)$
      $t_k = \underset{i \in \mathbf{a}_k}{\mathtt{max}} \left(t_w(i,k) + t_{ETA}(i,k) \right)$
    end for
    $J = - \underset{l \in \mathcal{K}}{\mathtt{max}}\, t_l$
  else
    $J = -\infty$
  end if
end procedure

can be calculated. By adopting the policy where the faster agent waits for the slower agent, the termination time of the task is the estimated time of the latest agent among the coalitions. The procedure of the score function and the entire process of the A-PSO are summarized in Algorithms 3.2 and 3.3, respectively.

Algorithm 3.3 A-PSO-based TA algorithm

procedure $[gbest, P_g^{N_s}]$ =A-PSO-BASED TA ALGORITHM$(\mathcal{I}, \mathcal{K}, N_s, N_i)$
  $\mathcal{S} \triangleq \{1, 2, ..., N_s\}$
  Initialize particles X_s^0, V_s^0, $P_s^0 = X_s^0$, $\forall s \in \mathcal{S}$
  Transform X_s^0 to path matrix $\mathbf{P}_s^0$, $\forall s \in \mathcal{S}$
  $pbest_s = SCORE(\mathbf{P}_s^0)$, $\forall s \in \mathcal{S}$
  for $n = 1 : N_i$ **do**
    for $s = 1 : N_s$ **do**
      Transform X_s^n to path matrix $\mathbf{P}$
      Particle refinement
      $J_s^n = SCORE(\mathbf{P})$
      if $J_s^n > pbest_s$ **then**
        $pbest_s = J_s^n$
        $P_s^n = X_s^n$
      else
        $P_s^n = X_s^{n-1}$
      end if
    end for
    $s^* = \underset{s \in \mathcal{S}}{\texttt{arg max}}\, pbest_s$
    $gbest = pbest_{s^*}$
    $P_g^n = P_{s^*}^n$
    Update particles using Eqs. (3.2) and (3.3)
  end for
end procedure

3.6 Task-based Particle Swarm Optimization : T-PSO

In the previous section, A-PSO is proposed to find an optimal solution using a metaheuristic approach. Because A-PSO follows the problem statement in Section 2.2, the particle was encoded as a path matrix $\mathbf{P}$. However, the number of particles not satisfying the constraints is still considerable despite the refinement procedure. To deal with this problem, task-based PSO (T-PSO) is proposed in this section.

The main idea of T-PSO is the modification of particle encoding, which follows the problem statement in Section 2.4. In T-PSO, the particle is encoded as a coalition vector $\mathbf{a}_k$ and precedence order vector $\mathbf{v}$. It can be stated that T-PSO solves the problem defined in Eqs. (2.15)–(2.16) instead of Eqs. (2.1)–(2.4). There exist two main differences between T-PSO and A-PSO with regard to the particle encoding and the particle refinement procedures.

3.6.1 Particle Encoding

Let X_s^n be the particle encoded for T-PSO, which represents coalition members and precedence order of tasks given by

$$X_s^n = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_M\ \mathbf{v}]^{\mathrm{T}},\quad \mathbf{a}_k \in \mathbb{N}^{z_k},\quad \forall k \in \mathcal{K},\quad \mathbf{v} \in \mathbb{R}^M \tag{3.10}$$

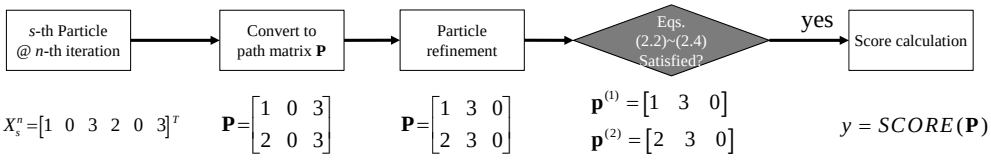
where $\mathbb{N}^{z_k}$ is the set of z_k–dimensional natural number vector. The precedence order of tasks is encoded as $\mathbf{v}$. The k-th element of $\mathbf{v}$ denotes a *precedence value* of task k. In this study, the smaller precedence value implies that the task would be performed sooner than other tasks having larger precedence value. When several precedence values are equal, the task with smaller index has the priority. For instance, $\mathbf{v} = [0.5, 0.2, 0.2]$ indicates a precedence order $2 \to 3 \to 1$.

This encoding has several advantages compared to that of A-PSO. First, the constraint of Eq. (2.10) is inherently satisfied because $\texttt{card}(\mathbf{a}_k) = z_k$. In addition, the DAG constraint of Eq. (2.13) is always satisfied because the precedence order of tasks is clearly determined by $\mathbf{v}$. The graph having a topological order is DAG. Finally, the dimension of the particle is $(\Sigma_{k=1}^{M} z_k + M)$, which is much smaller than that of the vectorized $\mathbf{A}$ and $\mathbf{V}$, $(NM + M^2)$.

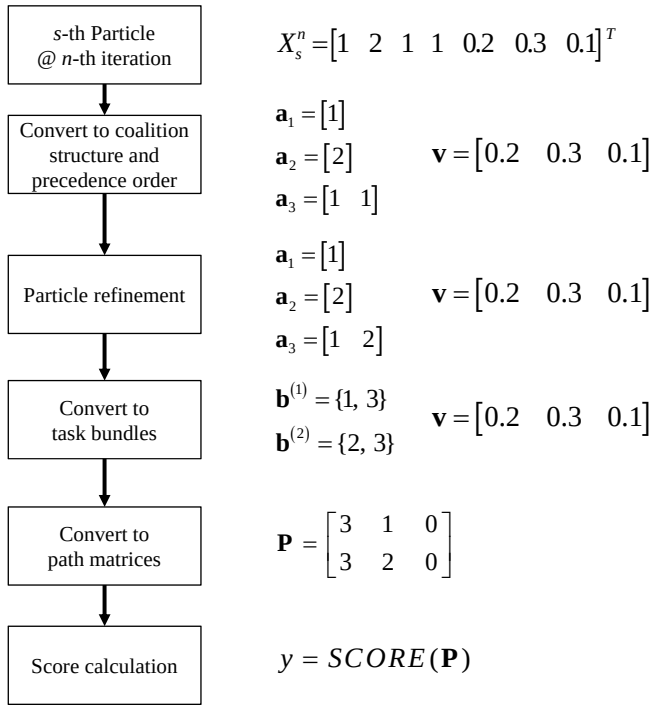
The score of the particle is calculated as A-PSO because the path matrix $\mathbf{P}$ can be uniquely determined by $\mathbf{A}$ and $\mathbf{V}$. Figure 3.6 summarizes the score calculation procedures of A-PSO and T-PSO, respectively, using an illustrative example.

3.6.2 Particle Refinement

The procedure of particle refinement in T-PSO is summarized as follows. First, if an coalition element of the position vector is outside the range of $[1, N]$, then the element is replaced by a random integer in $[1, N]$. Second, if there exist duplicated elements among $\mathbf{a}_k$, all elements in $\mathbf{a}_k$ are replaced by a random permutation. Finally, if a particle does not satisfy the constraint of Eq. (2.16), then the particle is entirely and randomly regenerated to meet the constraint of Eq. (2.16).



(a) **A-PSO**



(b) **T-PSO**

Figure 3.6 Score calculation procedures of A-PSO and T-PSO

3.7 Numerical Results

A SEAD mission is considered to investigate the performance of the five proposed algorithms: i) exact algorithm, ii) A-SGA, iii) T-SGA, iv) A-PSO, and v) T-PSO. To compare the performances of the algorithms, computation time and mission completion time are evaluated for various test problems. The configuration of the test problem is characterized by the number of UAVs $N \in \{1, 2, ..., 10\}$, the number of tasks $M \in \{1, 2, ..., 10\}$, and the length in kilometers of a side of the square mission area $D \in \{100, 200, 300\}$ [76]. That is, 300 different configurations are considered. For instance, problem 'N10M10D300' indicates that there are ten UAVs and ten tasks in the square mission area with a side of 300 km as shown in Fig. 3.7.

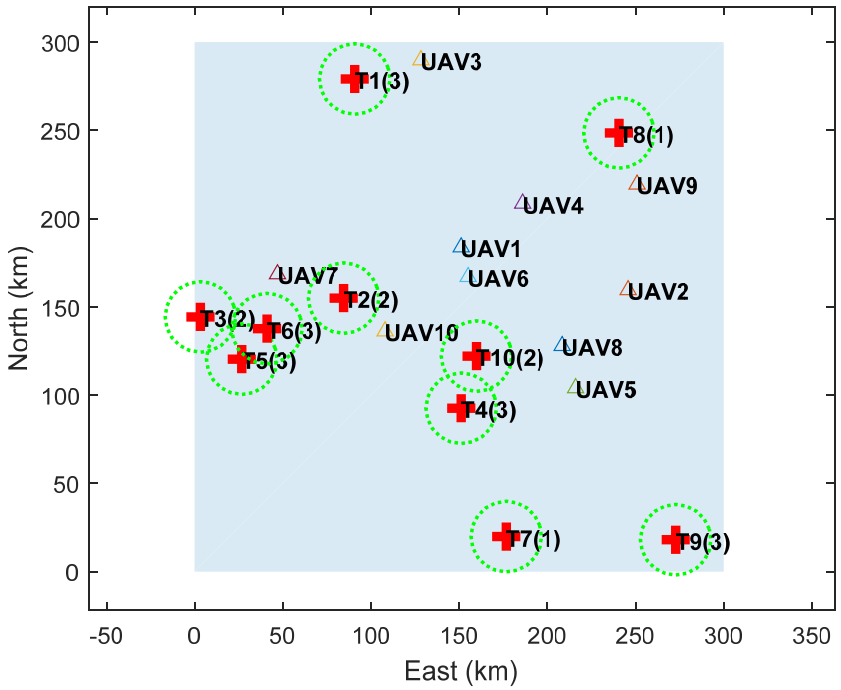


Figure 3.7 Example scenario of the SEAD mission (N=10, M=10, D=300)

The number inside the parenthesis in Fig. 3.7 denotes the number of required UAVs. The dotted circle around a task is related to the safe range, i.e., the radius of which indicates the detection range of the radar, which is set to 30 km. In this study, it is assumed that every UAV moves in 2–dimensional space with a speed of 200 m/s, and collision between UAVs is not considered. It is also assumed that the task execution time at the task is relatively short for the SEAD mission, t_w is set to zero in this study.

For each of the 300 configurations, 100 random test problems are generated where the positions of UAVs and targets are randomly chosen. Also, the required number z_k of UAVs for task k is randomly selected as $1 \leq z_k \leq min(N, 3)$. To ensure the existence of the solution, the maximum number $y^{(i)}$ of tasks allowed for agent i is randomly selected as $1 \leq y^{(i)} \leq max(M, 3)$.

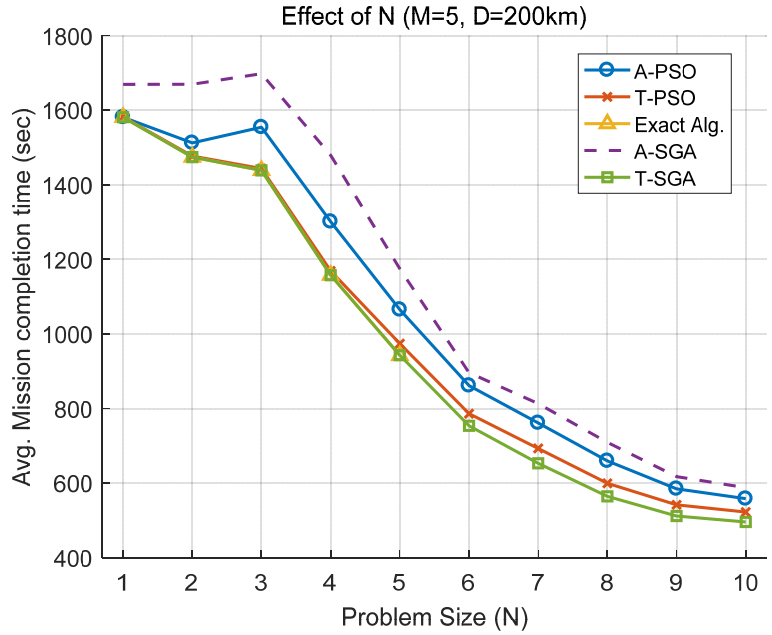
Simulation is performed using a desktop computer with an Intel Core i5-4670 @ 3.40 GHz with 16 GB of RAM and MATLAB on a Windows 7 operating system. Moreover, `parfor`-loops in the Parallel Computing Toolbox of MATLAB is utilized to speed up the computation of the exact algorithm. Populations of particles for A-PSO and T-PSO are set as 500 and 120, respectively. T-PSO requires less particles than A-PSO because a particle of T-PSO satisfies several constraints inherently. Initial positions of 1% of the total population are set as the solution of A-SGA so that both PSOs have at least same performance with A-SGA. For both PSOs, the maximum iteration number N_i is set to 1,000. As an additional stop condition of PSOs, the maximum stall number N_{stall} is set to $min(max(15N_d, 100), 500)$ where N_d is the dimension of a particle. That is, the algorithm stops if the best fitness function value are not improved during the recent N_{stall} generations. PSO parameter ω is set to 1.0, and $c_1 = c_2 = 2.05$ and $\chi = 0.729$ in Eq. (3.4).

To investigate the effect of N, the averaged mission completion time and

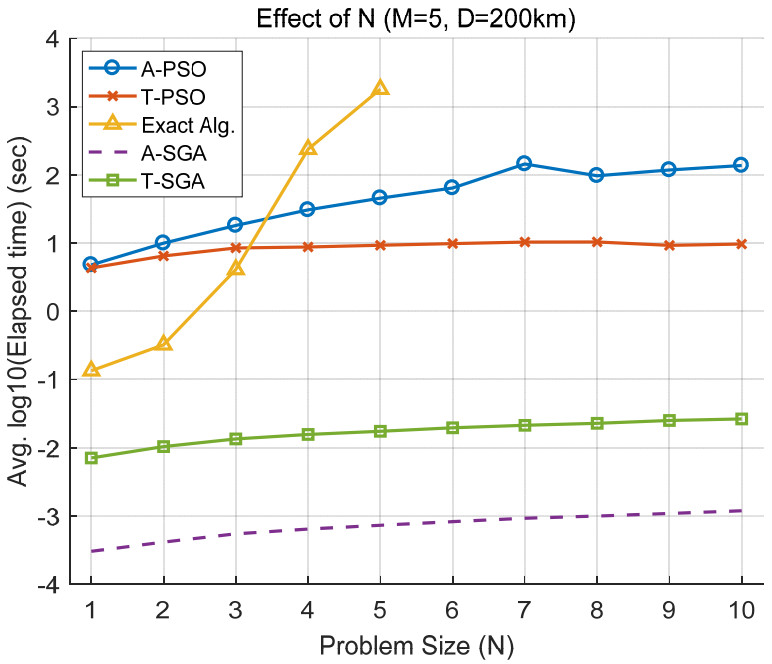
the log-scaled average computation time through 100 trials are shown in Fig. 3.8. M and D are set to 5 and 200, respectively. Considering the averaged mission completion time, T-SGA shows almost same performance with the exact algorithm and outperforms the other methods. The performance of T-PSO is better than A-PSO and A-SGA. However, the performance gap between different methods decreases as N increases. In view of average computation time, A-SGA uses the shortest time while the exact algorithm spends much time as N increases. Note that A-PSO is more sensitive to N than T-PSO, because the particle's dimension of A-PSO is NM (Eq.(3.5)) whereas that of T-PSO is $\Sigma z_k + M$ that is independent of N (Eq. (3.10)).

Figure 3.9 shows the results of another test problems, which are solved to investigate the effect of M. N and D are set to 5 and 200, respectively. As M is growing, the average mission times tend to increase proportionally. Similar to the previous results on the effect of N, T-SGA shows the best performance and T-PSO is better than A-PSO and A-SGA. Note that the performance gap increases as M increases. On the other hand, the exact algorithm and T-SGA require much computation time as M increases, because they consider the candidates of $M!$ combinations. T-PSO, A-PSO, and A-SGA are still less sensitive to M than other methods.

Figure 3.10 shows the effect of D where N and M are both set to five. It can be stated that D is an density index of the mission environment when N and M are constant. As D increases, the average mission time increases proportionally, because the speed of UAVs is constant. Also, the performance gap increases as D increases. Note that D does not influence the computation time when N and M are fixed.

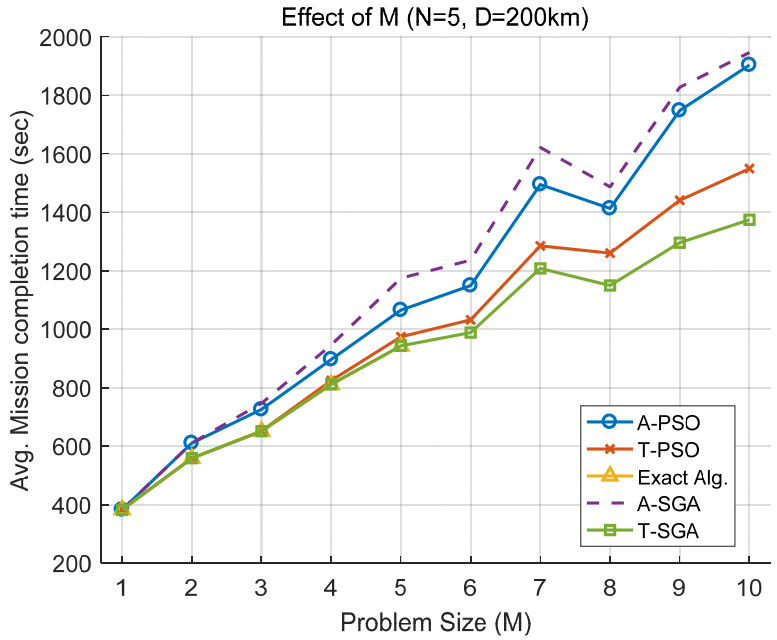


(a) Average mission completion time with respect to N (M=5, D=200)

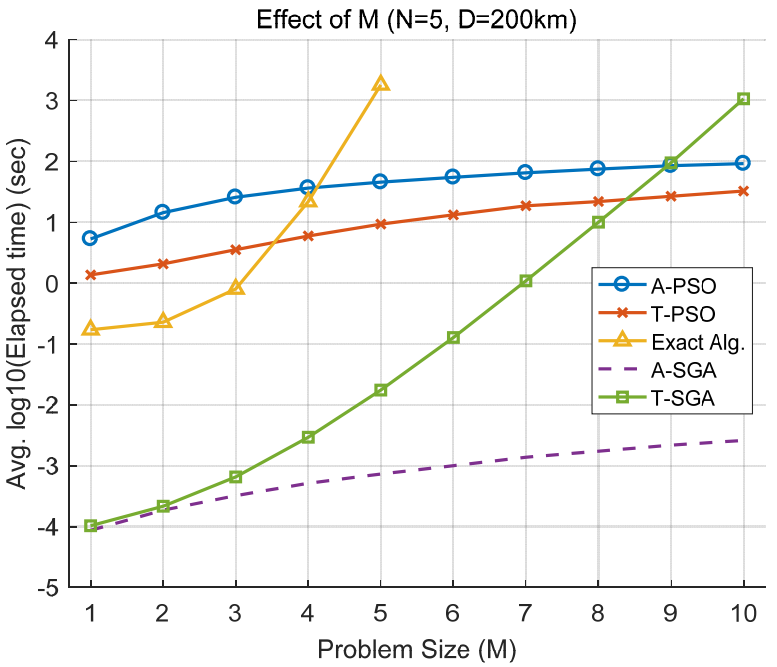


(b) Average elapsed time with respect to N (M=5, D=200)

Figure 3.8 Comparison of centralized task allocation schemes along N

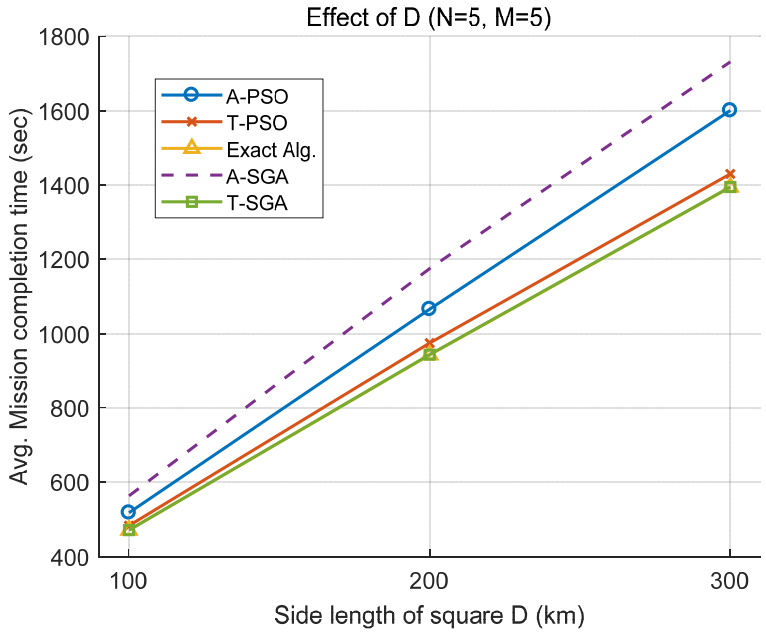


(a) Average mission completion time with respect to M (N=5, D=200)

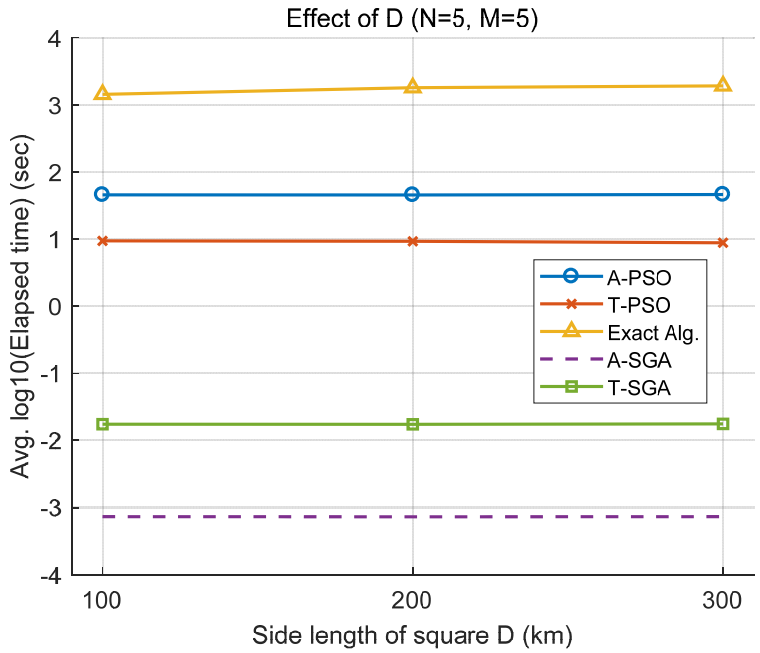


(b) Average elapsed time with respect to M (N=5, D=200)

Figure 3.9 Comparison of centralized task allocation schemes along M



(a) Average mission completion time with respect to D (N=5, M=5)



(b) Average elapsed time with respect to D (N=5, M=5)

Figure 3.10 Comparison of centralized task allocation schemes along D

In Table 3.1, the tendencies of mission completion time and computation according to N, M, and D are summarized. Also, the recommended algorithms and corresponding reasons for different types of problems are presented in Table 3.2.

Table 3.1 Summary of the numerical results

	Mission completion time	Runtime
N	As N increases, mission time becomes shorter and performance gap becomes smaller	Runtimes of exact algorithm and A-PSO increases explicitly as N increases
M	As M increases, mission time becomes longer and performance gap becomes larger	Runtimes of exact algorithm and T-SGA increases explicitly as M increases
D	As D increases, mission time becomes longer, performance gap becomes larger	The effect of D on computation time is very small

Table 3.2 Conclusion on the centralized task allocation methods

Mission Environment		Recommendation	Reason
High density		A-SGA	· Performance gap is small · Fast algorithms are preferred
Low density	Small No. of tasks	T-SGA	· Sensitive to M · Nearly optimal performance
	Large No. of tasks	T-PSO*	· Less sensitive to N and M

*The performance of T-PSO is usually better than or at least same as that of A-SGA

The performances of the proposed algorithms can be summarized by analyzing the simulation results. First, there exists a close relationship between deterministic solvers. Whereas the exact algorithm considers all possible cases of visiting order and coalition members, A-SGA decides tham in a greedy manner. To relax the computational burden of the exact algorithm and to improve the efficiency of A-SGA, only coalition members are greedily selected in T-SGA.

As a result, T-SGA showed the same performance with the exact algorithm using less computation time. Second, metaheuristic methods depend on problem formulations. Since the task-based formulation has reduced constraints and relaxed integer conditions, T-PSO provided better performance than A-PSO.

Chapter 4

Distributed Task Allocation

This chapter provides distributed algorithms to solve the TA problem where the master control center plays a minimal role and agents allocate tasks themselves. Two distributed TA algorithms are proposed based on market-based control paradigm: project manager-oriented and task-oriented methods. After analyzing the properties of convergence and scalability, baseline algorithms for a connected network are extended to deal with time-varying network topology including isolated sub-networks due to a limited communication range. To demonstrate and compare the performance of the proposed algorithms, numerical simulations for a SEAD mission are conducted.

To consider completion time as well as the priority between tasks, the objective function J and the score function s_k in Eq. (2.1) are redefined as follows,

$$J = \sum_{k=1}^{M} s_k(t_k(\mathbf{P})) \tag{4.1}$$

$$s_k(t_k(\mathbf{P})) = w_k e^{-\lambda_k (t_k - t_k^0)} \tag{4.2}$$

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 the task k is generated.

4.1 Assumptions

Throughout this chapter, the following assumptions are considered.

Assumption 4-1. 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.

Assumption 4-2. The network topology does not change during the process of TA.

Assumption 4-3. 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, for extended algorithms dealing with dynamic environment, these two assumptions will be omitted. The last assumption may not be appropriate for real 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.

4.2 Project Manager-oriented Coalition Formation Algorithm : PCFA

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 agent to allocate the given tasks by themselves based on the information given by communications between agents.

Suppose that the tasks and the agents are interpreted as the projects and the contractors [48]. 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. 4.1, where *fitness* and *resumé* are scalar values representing quantitative suitability of agents. The four-phase algorithm repeats until all tasks are assigned. 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 $\mathbf{x}^{(i)}$, average speed $v^{(i)}$, and the winning advertisement vector $\mathbf{A}^{(i)}$. Table 4.1 summarizes the usages of local variables with an example.

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: $\mathbf{T}_p(k)$ $(=\mathbf{x}_k)$, $\mathbf{T}_a(k)$, $\mathbf{T}_m(k)$ $(=z_k)$, $\mathbf{T}_w(k)$ $(=w_k)$, $\mathbf{T}_\lambda(k)$ $(=\lambda_k)$, and $\mathbf{T}_0(k)$ $(=t_k^0)$. The variable $\mathbf{T}_a(k) = 1$ if the task k is assigned to some agents, and 0 otherwise.

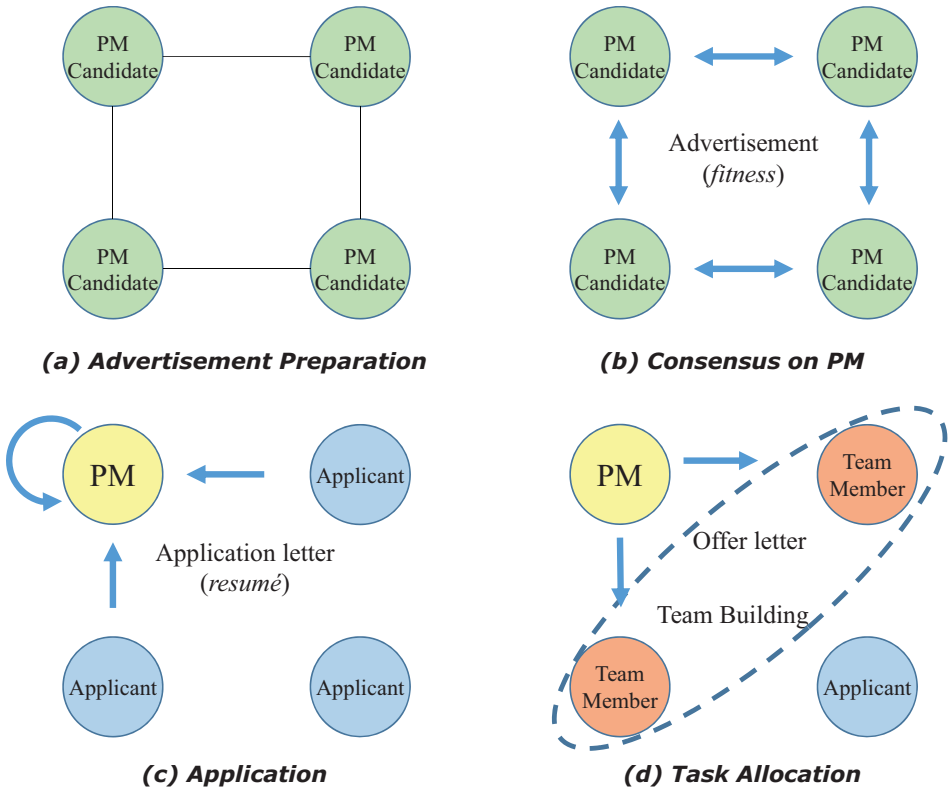


Figure 4.1 Task allocation procedures in PCFA. Broadcasting messages can be transferred when the network between the agents is connected, where solid line in (a) denotes the connectivity between two nodes. In this topology, direct communications between diagonal agents are not available

Table 4.1 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	$\mathbf{x}^{(i)} = [10\ 100]$	Position of agent i: [10 100] (m)
average speed	$v^{(i)} = 10$	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

Phase 1: Advertisement Preparation

At the first phase, the agent i prepares a winning advertisement vector $\mathbf{A}^{(i)}$ as summarized in Algorithm 4.1. 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

$$f^{(i)}(k) = s_k(t_{ETA}) = w_k e^{-\lambda_k(t_{ETA}+t_w-t_k^0)} \tag{4.3}$$

Each agent does not know the path list $\mathbf{P}$ of the all agents, and therefore agents cannot calculate the exact score function. Instead, the approximated score in Eq. (4.3) 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. (2.7) 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, $p_{i,1}$ cannot be equal to k. Additionally, the time t, when t_{ETA} is calculated, should be considered because the evaluation of t_{ETA} may be necessary during the mission due to pop-up task. The modified definition of t_{ETA} for the distributed TA procedure is as follows,

$$t_{ETA}(i,k,t) = \begin{cases} t + \left\| \mathbf{x}_k - \mathbf{x}^{(i)} \right\| / v^{(i)}, & \text{if } \ p_{i,1} = 0 \\ t_j + \| \mathbf{x}_k - \mathbf{x}_j \| / v^{(i)}, & \text{otherwise} \end{cases} \tag{4.4}$$

In PCFA, advertisement for a certain task is only allowed to the 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. 4.1, 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 4.1 line 5). Note that $Z_{max}^{(i)} (= n(\mathcal{N}^{(i)}) + 1)$ 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 $\mathbf{A}_{prev}^{(i)}$, which is designed to be shared through communications between neighboring agents in the following phase. When the agent i fails to calculate the *fitness*, it generates a dummy winning advertisement (Algorithm 4.1, line 14).

Algorithm 4.1 Phase 1 for agent i.

procedure ADVERTISEMENT PREPARATION(i, t)
  $f^{(i)} = \mathbf{0}_{n(\mathcal{K})\times 1}$
  $Z_{max}^{(i)} = n(\mathcal{N}^{(i)}) + 1$
  for $k = 1 : n(\mathcal{K})$ **do**
    if $(\mathbf{T}_a(k) = 0)$ & $(Z_{max}^{(i)} \geq z_k)$ & $(n(\mathbf{p}^{(i)}) < y^{(i)})$ **then**
      Calculate $f^{(i)}(k)$ ▷ Eq. (4.3)
    end if
  end for
  if $\sum_{k=1}^{n(\mathcal{K})} f^{(i)}(k) \neq 0$ **then**
    $k^* = \arg\max_{k\in\mathcal{K}} f^{(i)}(k)$
    $q^* = f^{(i)}(k^*)$
    $\mathbf{A}^{(i)} = [i, k^*, q^*]$
  else
    $\mathbf{A}^{(i)} = [i, 0, 0]$
  end if
  $\mathbf{A}_{prev}^{(i)} = \mathbf{A}^{(i)}$
end procedure

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 4.2. Note that every agent prepares two advertisement vectors in phase 1, i.e., i) previous winning advertisement vector $\mathbf{A}^{(i)}_{prev}$, and ii) 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 $\mathbf{A}^{(j)}_{prev}$, then $\mathbf{A}^{(i)}$ 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 path between two vertices [67]. 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 distributed agents may not be able to recognize the exact topology of the network because the communication connection between agents may be changed during the mission. Thus, the network diameter for the worst case, $N-1$, is selected as a conservative limit (Algorithm 4.2 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 4.2∼4.3 where $\mathbf{A}^{(i)}(j)$ denotes the j-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 $\mathbf{A}^{(i)}$. $\mathcal{N}^{(i)}(j)$ denotes the j-th element of $\mathcal{N}^{(i)}$. Note that tie-break rule is applied by prioritizing agent with a lower index (Algorithm 4.3 lines $7 \sim 9$).

Algorithm 4.2 Phase 2 for agent i.

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

Algorithm 4.3 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
$\mathbf{A}^{(i)}_{prev} = \mathbf{A}^{(i)}$

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.,

$$r^{(i)}(k,t) = t_{ETA}(i,k,t). \tag{4.5}$$

where $t_{ETA}(i,k,t)$ is defined in Eq.(4.4). 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. 4.1. Thus, the PM should compete with other applicants to be a team member. Phase 3 is summarized in Algorithm 4.4, where the left arrow operator, $\Leftarrow$, augments the right row vector into the left matrix of the arrow operator.

Algorithm 4.4 Phase 3 for agent i.

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

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 4.5, 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 4.5 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

4.3 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 be utilized for cooperative timing missions even if the network is not fully connected. However, 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. 4.1(a), the task cannot be accomplished because the maximum number of neighborhood is two. The motivation of TCFA is to handle this limitation. By relaying the 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 4.2 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, all application letters are *radially propagated* from each agent (vertex) to its neighboring agents (adjacent vertices) by using one broadcasting. From the fact that the

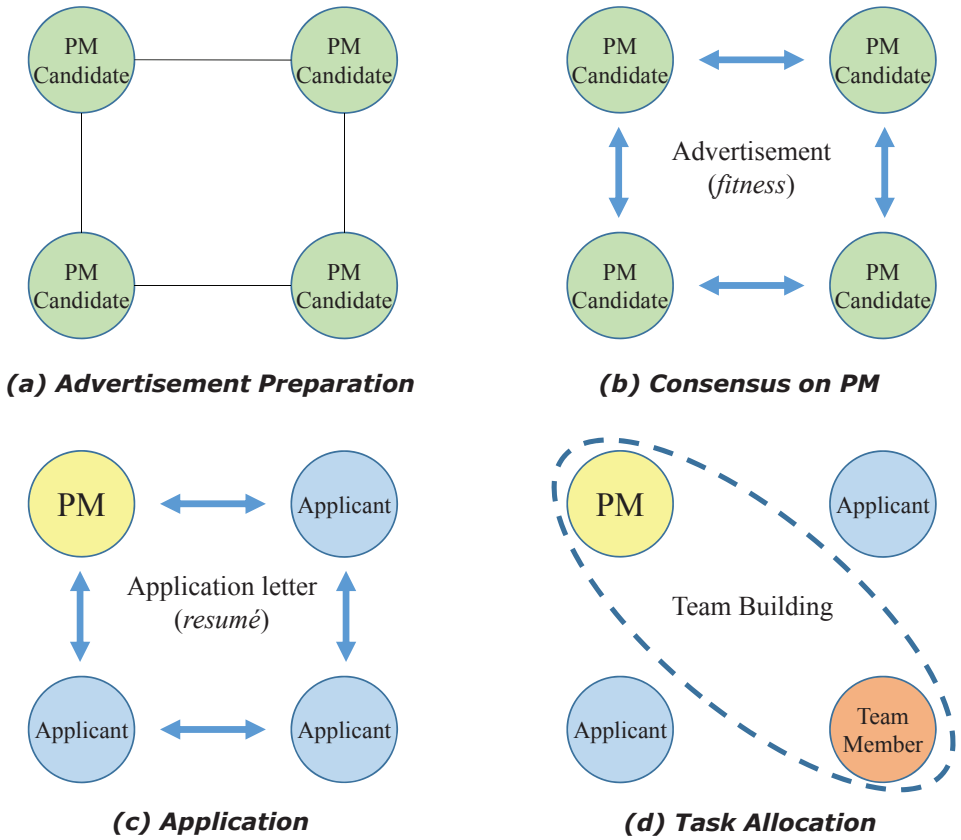


Figure 4.2 Task allocation procedures in TCFA. Broadcasting message can be transferred when the network between the agents is connected, where solid line in (a) denotes the connectivity between two nodes. In this topology, direct communications between diagonal agents are not available

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 are 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 perform task k, which means that z_k-1 members should 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 Algorithms 4.6∼4.8. When the agents have identical *resumé* values, the agent with the lowest index is selected.

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 4.9.

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 and computational loads.

Algorithm 4.6 Modified Phase 3 for agent i.

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

Algorithm 4.7 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 4.8 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

Algorithm 4.9 Modified Phase 4 for agent i.

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

4.4 Modified Greedy Distributed Allocation Protocol : Modified GDAP

For a comparative study, greedy distributed allocation protocol (GDAP) [41] is modified and adapted to the target problem. The coalition formation scheme, including the advertising and applying processes of GDAP, is similar to the proposed TA algorithms, but different aspects will be explained in the subsequent section dealing with the 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 their own resources as an initial condition. The information of each task is randomly distributed to each agent at the beginning of the TA process, and the agents who received the information are called the managers of each task. If the network topology between agents is not fully connected, then only the manager's neighboring agents are permitted to contribute to the task. In other words, not all agents obtain the information about all tasks. All agents, including managers, are called contractors. Each manager finds contractors to work with, 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 the target problem, GDAP algorithm is modified as follows. In phase 1, agent i, who is a manager of several tasks, calculates its *fitness* defined in Eq. (4.3) 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 4.1. 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 the manager has sufficient applicants, the manager selects the best team members according to the *resumé* value. In this study, 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. 4.3.

Note that GDAP needs less communication and does not require iterations due to the lack of consensus process. In dynamic environment, rapid decision making may enhance the efficiency of the TA. Also, in distributed 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.

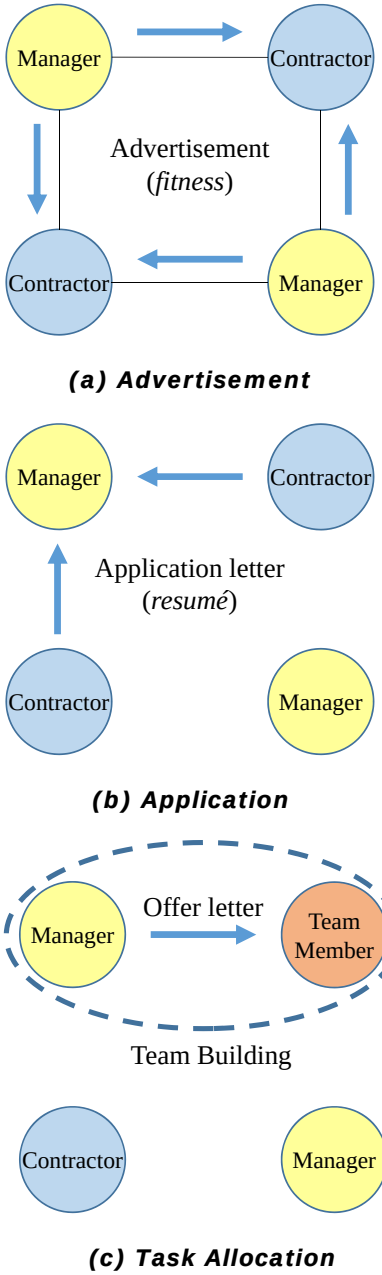


Figure 4.3 Task allocation procedures in GDAP. Broadcasting message can be transferred when the network between the agents is connected, where solid line in (a) denotes the connectivity between two nodes. In this topology, direct communications between diagonal agents are not available

4.5 Properties

4.5.1 Convergence

In this section, it is proven that a conflict-free solution generated by the proposed TA algorithms converges. *Convergence* of a TA algorithm means 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. (2.2)∼(2.4). Theorem 4.1 shows the convergence of PCFA in a connected network.

Theorem 4.1. *Consider PCFA and given tasks with involved agents. Let us assume that the maximum number of required 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. Theorem 4.1 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 $N - 1$. If the *fitnesses* of different PM candidates are the same, then the candidate with lower agent index is selected. This procedure is applied similarly for the case that 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 in phase 1, 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 number of necessary agents for the given tasks is bounded by the number of maximum available agents, M tasks will be assigned during M rounds within a finite time. Moreover, because each task is augmented to the end of the existing path vector, the resulting path vector $\mathbf{P}$ satisfies the DAG constraint of Eq. (2.4). □

According to Theorem 4.1, the convergence of TCFA can be easily shown. Phase 1 and phase 2 of TCFA do not form a bottleneck, as in the proof of Theorem 4.1. 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.

4.5.2 Scalability

Communication

The amount of communication for TA is analyzed in this section. As shown in Lemma 4.1, for a connected network, there exists a polynomial upper bound for the number of communications which are required for allocating given tasks. It is assumed that the agents do not know the diameter of the current network. Note that one broadcast of an agent to its neighborhood is counted as one communication. It can be stated that PCFA is scalable to large problems with regard to the amount of communication.

Lemma 4.1. *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.*

$$C_{TA}(N, M) = 2M(N-1) \tag{4.6}$$

$$B_{TA}(N, M) = 12M(N-1) \tag{4.7}$$

Proof. By Theorem 4.1, M tasks are allocated within M rounds. Now, let us estimate 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 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 therefore the upper bound is $2M(N-1)$ for M tasks. 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 $12M(N-1)$ bytes. □

The number of communications required for TCFA can be computed as in Lemma 4.1. 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.

Time Complexity

The proposed algorithms are scalable to large-sized problems in terms of time complexity. Theorem 4.2 shows that PCFA runs in a polynomial time.

Theorem 4.2. *The asymptotic worst-case time complexity of the PCFA with M tasks and N agents can be expressed as follows,*

$$T_{TA}(N,M) = O(M^2 + MN^2 + MN\log(N)) \tag{4.8}$$

Proof. By Theorem 4.1, 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 4.1 line 4 is $O(M)$ because the ETA is calculated for M times. The *if-statement* in line 9 of Algorithm 4.1 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, which 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 known that the time complexity of the sorting is $O(n\log(n))$, where n is the number of elements to sort [69]. 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 $O(3M+(N-1)^2+N\log(N))$. Hence, 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 4.2. 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. To delete 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 follows,

$$T_{TA}(N, M) = O(M^2 + MN + MN^2 + MN^2\log(N) + MN^4) \tag{4.9}$$

4.5.3 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 distributed 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 required calculations are composed of fundamental arithmetic operations or logical operations such as 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 that the *fitness* and *resumé* are defined. Proof of convergence can be equivalently applied to various cooperative TA problems regardless of the definition of *fitness* and *resumé*. In addition, any impact-time-control guidance law can be integrated as a low level controller, because the proposed TA algorithm only decides the sequence of the path list and corresponding time table.

4.5.4 Comparison with GDAP

There exists an significant difference between the proposed algorithms and the GDAP. While the selection of the auctioneer is negotiated for each round in the proposed methods, all auctioneers are randomly chosen in the GDAP. As a result, different TA solutions are provided 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 the 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 the 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 may improve the efficiency. For example, suppose a dynamic environment has several disconnected sub-networks with a limited communication range. In this case, if the manager of a certain task is too far from the corresponding task, the GDAP will form an inefficient coalition.

The GDAP has strong points with respect to less rounds and communication, and it can allocate multiple tasks during one round even in the connected network. However, a conflict may occur in the GDAP when the network is not *fully-connected*. In the fully-connected network, all pairs of vertices are directly connected. Let us consider an example shown in Fig. 4.4. Suppose that a manager broadcasts an advertisement of itself in phase 1. 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 simultaneously assigned to two different teams. To resolve this conflict, more cross-checking is necessary.

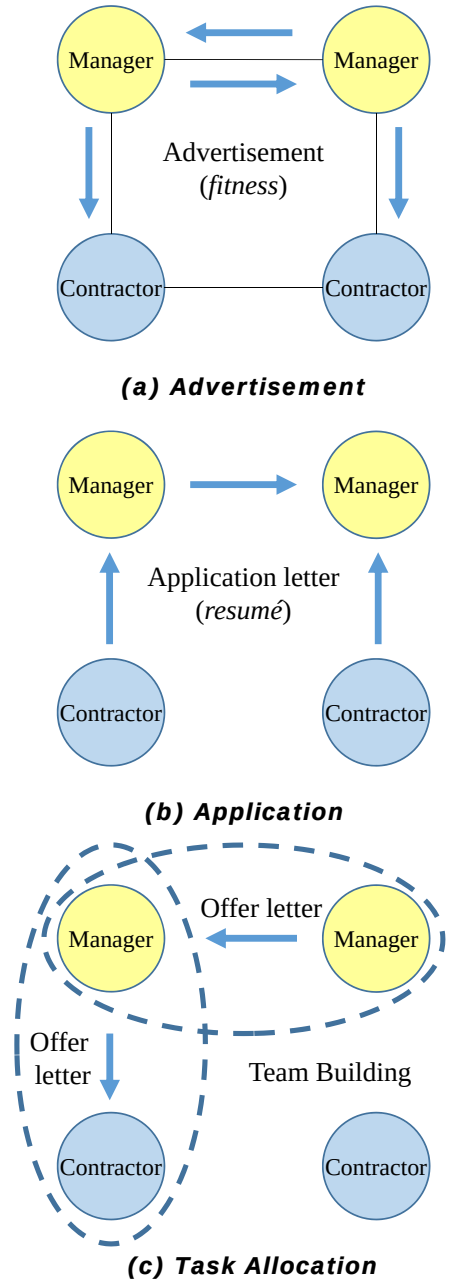


Figure 4.4 Possible conflict in GDAP. Broadcasting message can be transferred when the network between the agents is connected, where solid line in (a) denotes the connectivity between two nodes. In this topology, direct communications between diagonal agents are not available

4.6 TA Algorithm in Dynamic Environment

4.6.1 Challenges in Dynamic Environment

In dynamic environment, additional tasks may be given to agents. Assuming that network connectivity depends on the relative distance between agents, the network topology can be changed or even may be disconnected due to the mobility of the agents. The proposed algorithms may not work properly in dynamic environment as intended, because a static and connected network during each TA round is assumed. Especially, a disconnection during consensus progress may cause a conflict, i.e., two disconnected subgroups may have different ideas about who the PM is. Also, a disconnection during the application phase may create a disagreement about who the team member is.

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

4.6.2 Assumptions

Throughout this section, the following assumptions are considered.

Assumption 4-4. Agents communicate with each other in a synchronous manner, i.e., each agent communicates with other agent according to the scheduled time table.

Assumption 4-5. There is a mission control center (MCC) that monitors all of the agents, and the MCC and all of the agent update the task information **T** mutually.

Assumption 4-6. The clocks of agents are synchronized.

Assumption 4-7. Each agent knows the list of agents in its sub-network.

4.6.3 Distributed TA Architecture in Dynamic Environment

In this section, the proposed TA algorithms are extended to treat the problem in dynamic environment. Let us consider that each agent has TA block for a high-level controller and guidance/control block for a low-level controller. Once TA block calculates the path list and corresponding time table, the low-level controller drives the vehicle to arrive at the target in time. Guidance and control block runs every T_c time-step, and TA block runs every T_d time-step, where T_d depends on communication bandwidth. Figure 4.5 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. In the phase requiring several iterations for consensus with neighboring agents, only one iteration is performed at each execution of TA block. For instance, it takes $(N-1)T_d$ seconds for phase 2 of PCFA. In this study, the phase token $K^{(i)}$ is adopted, which indicates the phase number to be executed. The agent i resets the phase token $K^{(i)}$ to one for the following two cases. First, if there are no unassigned tasks, agents do not have to do TA process and reset the phase token to one. Another case is that members of agent i's sub-network are changed during the mission. Note that the proposed algorithm requires synchronous phase scheduling in each sub-network, and therefore, all members of the sub-network 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 sub-network. But, the information of the network topology is not used yet. In phase 3, the *resumé* should be changed because there exists a time gap between the application time and the team building time. Simultaneous arrival may fail if this gap is neglected. Therefore,

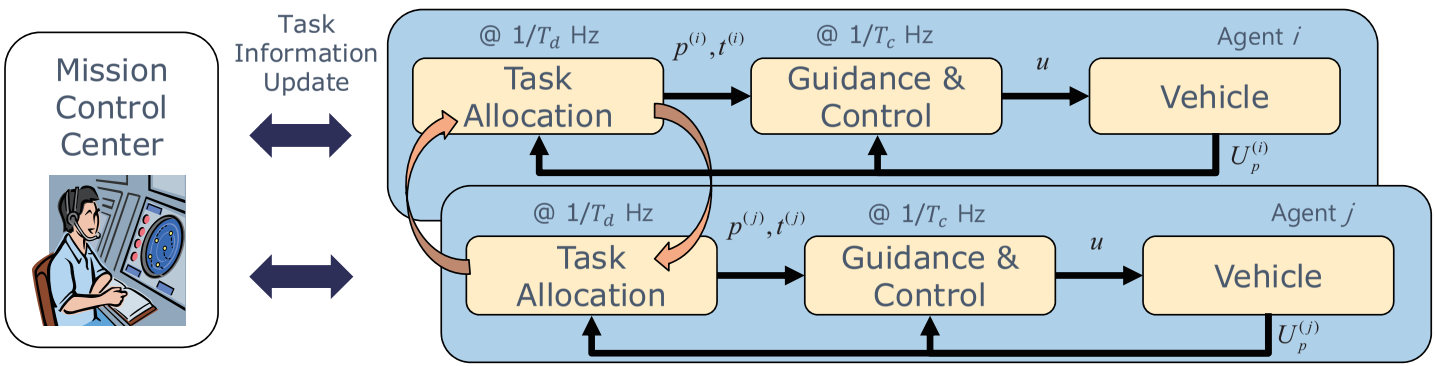


Figure 4.5 Real-time distributed TA architecture

the *resumé* of PCFA is modified as follows,

$$r^{(i)}(k,t) = t_{ETA}(i,k,t) + 2T_d \tag{4.10}$$

where $2T_d$ compensates for the time gap as well as the moving distance of the agent i during T_d time-step. The entire process of PCFA dealing with dynamic environment is presented in Algorithm 4.10.

Similarly, TCFA can be extended to deal with the dynamic environment. Agents prepare the application letter in the first run of phase 3 and then start making a consensus in the second run. The *resumé* of TCFA is modified as

$$r^{(i)}(k,t) = t_{ETA}(i,k,t) + 2(n_s^{(i)} - 1)T_d \tag{4.11}$$

where $2(n_s^{(i)} - 1)T_d$ compensates for the time gap as well as the moving distance of agent i during $(n_s^{(i)} - 1)T_d$ time-step. The modified pseudocode of case 3 for TCFA is presented in Algorithm 4.11 where $F_3^{(i)}$ is initialized to zero at the case 1 of TCFA.

Algorithm 4.10 TA Block for PCFA (i)

if There exist unassigned tasks **then**
    $n_s^{(i)}$ = number of nodes of sub-network
    Update location of RP
    if nodes of sub-network are changed **then**
        $K^{(i)} = 1$
        $\mathbf{L}_{app}^{(i)} = \emptyset$
    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 4.11 Case 3 for TCFA

1: **case** 3
2: **if** $F_3^{(i)} = 0$ **then**
3: PrepareApp(i)
4: **else**
5: ConsensusApp(i)
6: **end if**
7: $F_3^{(i)} = F_3^{(i)} + 1$
8: **if** $F_3^{(i)} = n_s^{(i)}$ **then**
9: $K^{(i)} = 4$
10: **end if**

4.6.4 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 environment, however, the number of members in the sub-network may not be sufficient to perform the given task due to a limited communication range. This problem can be resolved by adopting the concept of *rally point* (RP) which is a designated place to visit when an agent does not have any tasks to perform. Agents around the RP are connected, and thus, they can be put into the TA process again.

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 may be a time-efficient candidate for the RP. During the SEAD mission, however, the preferable location of RP may be on the safe border of the dangerous region and concurrently close to the ally's base.

The RP determination law should provide the same location of RP to all agents without using agents' positions, because the precise position information of agents is hard to be obtained. To derive the adaptation law for the determination of the RP position in SEAD mission, convex hull and Minkowski sum [77] 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 the 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. 4.6, the closest point from the base among the convex hull of the Minkowski sum is selected as the RP.

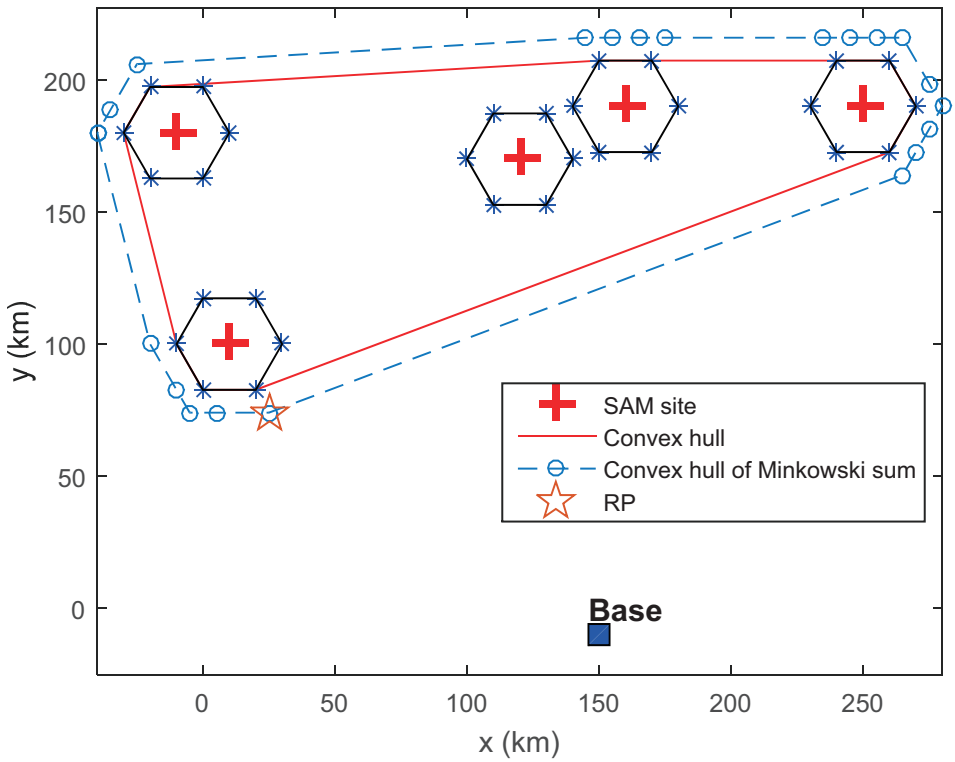


Figure 4.6 Choice of rally point

4.6.5 Convergence

In this subsection, convergence of the proposed algorithms in dynamic environment is analyzed. In this section, the *convergence* means that all tasks can be allocated to the agents within a finite time. Let us remind the assumptions in Section 4.1; 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 used, which reset the token when members of sub-network are changed. The concept of RP is adopted for the static and connected network. By gathering the agents not having any tasks around the RP, the static and connected network can be achieved. Therefore, according to Theorem 4.1, convergence of the proposed algorithms are guaranteed in the dynamic environment.

4.6.6 Deletion of Duplicated Allocation

Disconnected sub-networks due to the limited communication range may yield duplicated allocations, because each sub-network does not have the information of the other sub-networks. In this study, it is assumed that mission control center resolves this problem by noticing the status of the completed tasks to the agents. When the mission control center receives the completion notice reported by the coalition who visited the task, then the MCC broadcasts the completeness of tasks as shown in Fig. 4.5. Note that this treatment may degrade the performance of TA because multiple coalitions may head to the same task until one coalition completes the task.

4.7 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. Also, the proposed algorithms are applied to the dynamic SEAD scenario, a primary application of this study. The simulation is executed using a desktop personal computer equipped with an Intel Core i5-4670 @ 3.40 GHz, and 16 GB of RAM. MATLAB on Windows 7 operating system is used.

4.7.1 Scalability

The scalability of the proposed algorithms is examined for a problem with a static and connected network. By Theorem 4.2, the proposed TA algorithms have polynomial time complexity, and therefore, the parallel runtime, which is obtained by dividing the total runtime by N [78], can be estimated by the time complexity formula. The estimated parallel runtime $\hat{t}_r$ can be obtained by adapting two unknown parameters for the simplified time complexity formula as

$$\hat{t}_r^c = \alpha_c M^2 + \beta_c M N^2, \tag{4.12}$$

$$\hat{t}_r^d = \alpha_d M^2 + \beta_d M N^2 \log(N), \tag{4.13}$$

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 the 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 $1 \leq N \leq 30$, which results in 900 different problem sizes. For each problem, 100 Monte Carlo simulations are performed, and therefore 90,000 different problem cases are generated. The initial positions of agents and 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 [67] when the second smallest eigenvalue of the Laplacian matrix is greater than zero. For 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 is chosen as the worst-case value. The 900 sets of the (N, M, t_r) are used to identify the unknown parameters, and the goodness of fit [79] is evaluated by using the normalized mean square error R^2.

Table 4.2 Estimated parallel runtime

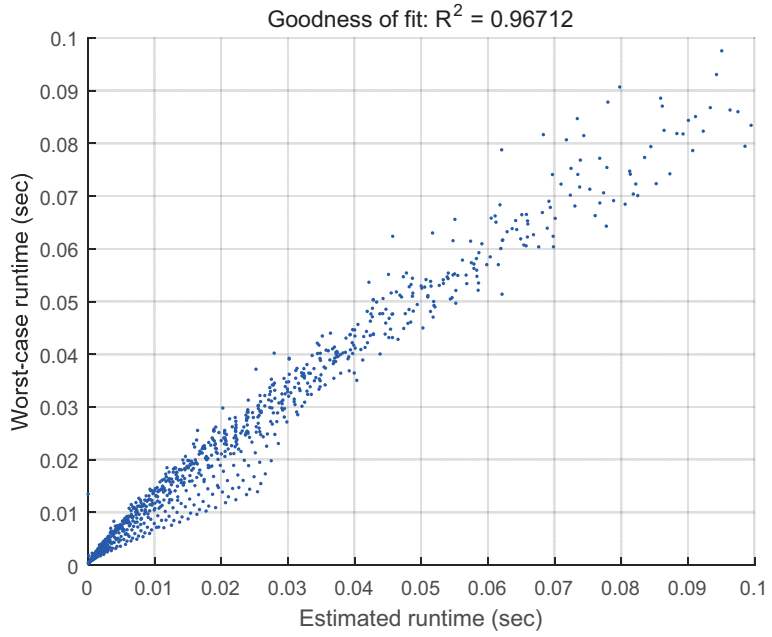
Formula	$\hat{t}_r^c = \alpha_c M^2 + \beta_c MN^2$		$\hat{t}_r^d = \alpha_d M^2 + \beta_d MN^2 \log(N)$	
Parameters	α_c	$2.7833 * 10^{-5}$	α_d	$1.9975 * 10^{-4}$
	β_c	$3.2584 * 10^{-6}$	β_d	$1.0549 * 10^{-5}$
Goodness of Fit	R^2	0.9671	R^2	0.9356

The identification results are summarized in Table 4.2, and Fig. 4.7(a) shows a comparison between $\hat{t}_r$ and t_r for each problem size. The estimated runtime shows good agreement with the worst-case runtime. Figure 4.7(a) shows that the

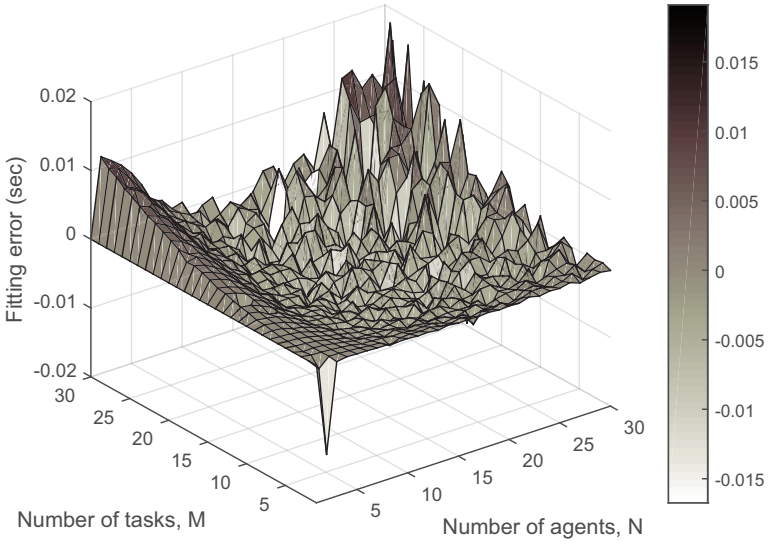
estimator reasonably represents the worst-case runtime. The difference between the estimated runtime and the worst-case runtime of PCFA and TCFA are shown in Fig. 4.7(b) and Fig. 4.8, respectively.

The polynomial time complexity was validated by using R^2 of the fitting results for a specific problem size. The simulation results show that the proposed TA algorithms can solve large-size problems. For example, for the case that N and M are both 30, the proposed algorithms solve the problem within one second. Note that network bandwidth was not considered in the simulation, and therefore, the presented runtime can be considered as 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 4.1. To take the worst case, the maximum number of communications is saved during the Monte Carlo simulations. Figure 4.9 shows the maximum communications with upper bound with respect to the problem size $N(=M)$. As derived in Eq. (4.6), the number of communications grows quadratically with the problem size. PCFA has some margin from the upper bound as shown in Fig. 4.9, because the number of offer letters is generally less than $N-1$. On the other hand, the number of communications for TCFA is same as the bound value. The reason is that $N-1$ iterations are performed in phase 2 and phase 3, as the network topology is assumed to be unknown.

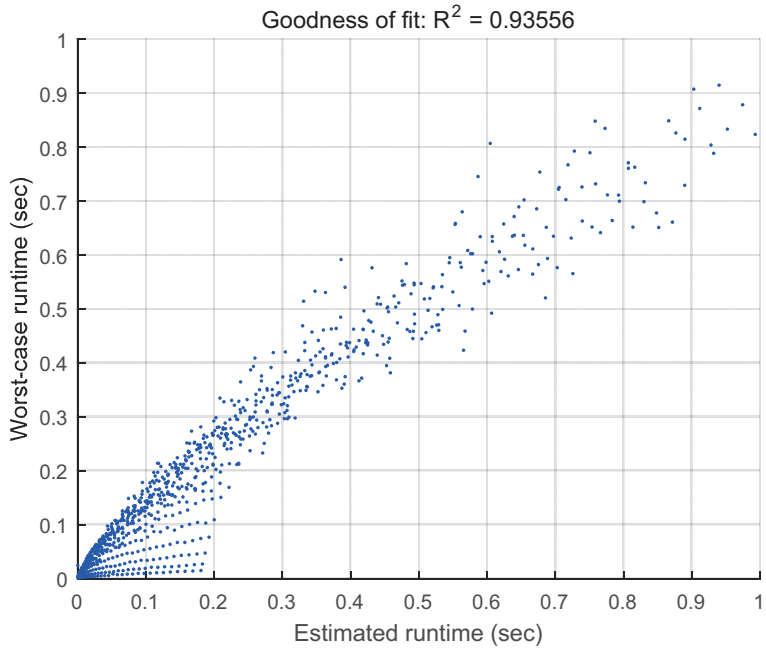


(a) **Runtime comparison**

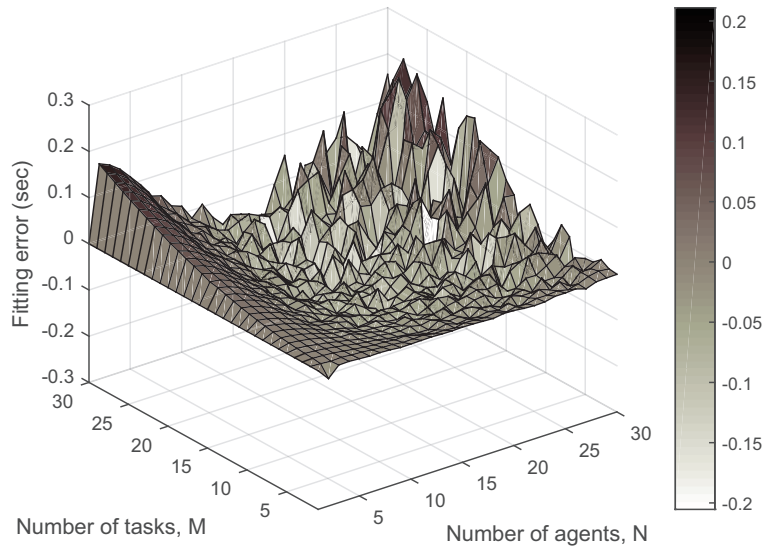


(b) **Estimation error of the worst-case runtime**

Figure 4.7 Parallel runtime estimation (PCFA)

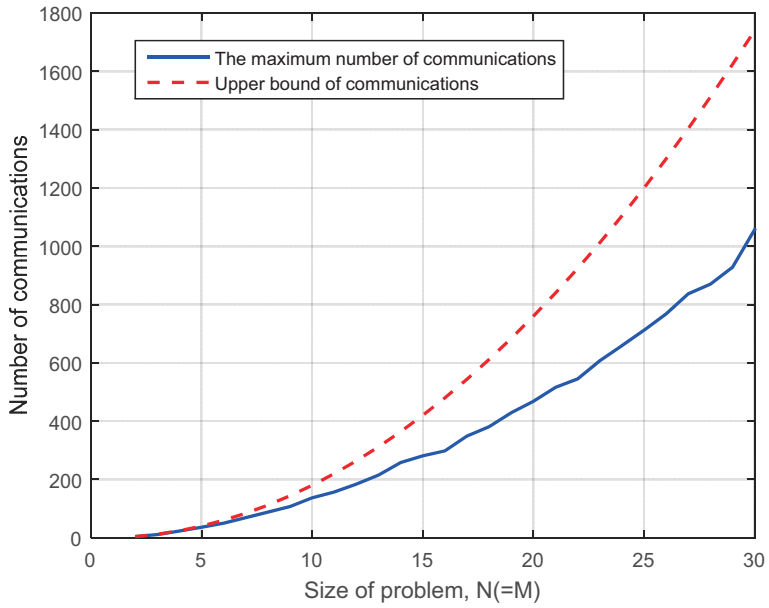


(a) **Runtime comparison**

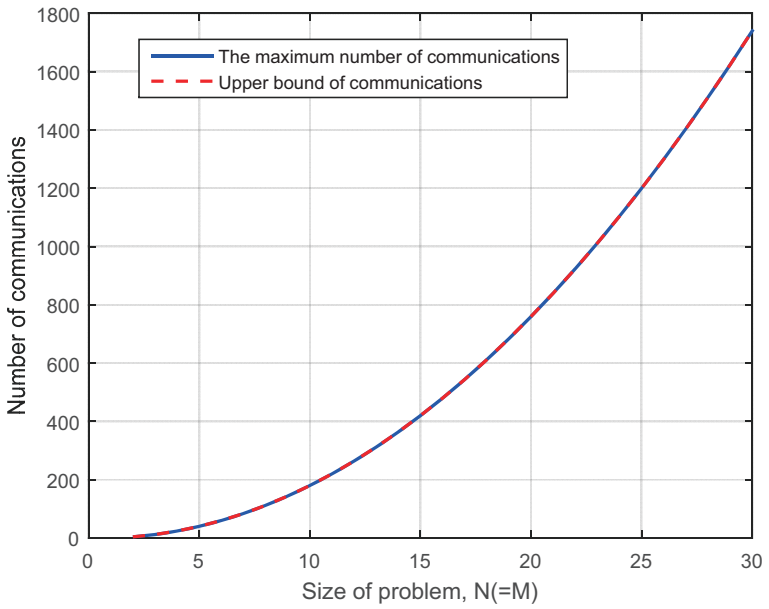


(b) **Estimation error of the worst-case runtime**

Figure 4.8 Parallel runtime estimation (TCFA)



(a) **PCFA**



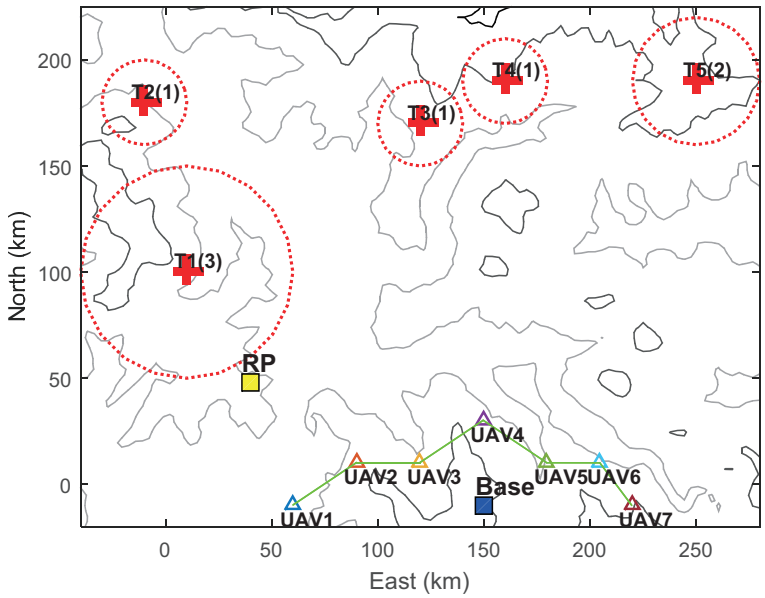
(b) **TCFA**

Figure 4.9 Effect of problem size on amount of communication

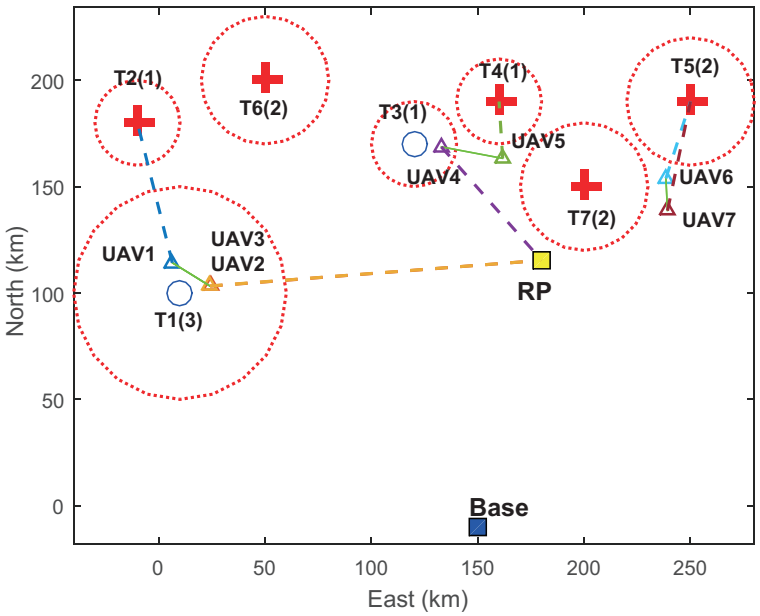
4.7.2 Application: SEAD Scenario

SEAD Environment

Figure 4.10(a) 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. 4.10, 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 4.10(b) shows the dynamic environment at 800 sec., where T6 and T7 are pop-up tasks and T1 and T3 are completed tasks. The dashed line connecting a UAV and a task is the remaining path of the UAVs.



(a) **Simulation snapshot at 0 sec**



(b) **Simulation snapshot at 800 sec**

Figure 4.10 SEAD environment

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 the team; thus, the scheduled time table and path list make the UAVs arrive at the common task simultaneously. If a UAV arrives at the given task earlier, then 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. It is assumed that task execution time at the target is relatively short for the SEAD mission, and therefore t_w is set to zero in this study. After the completion of the task, each UAV moves 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 all the tasks of the mission, they return to the base.

TA Results in Dynamic Environment with Various Communication Ranges

In this section, the performance of the proposed algorithms is analyzed with respect to various communication ranges through Monte Carlo simulations.

Detailed TA Progress of Sample Scenario

Detailed TA progress by TCFA is shown in Table 4.3. Table 4.3 shows TA results at some specified times for the sample scenario in Fig. 4.10(b) where the description on each variable can be found in Table 4.1. In the scenario, 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. (4.3), the inherent worth w_k of task k denotes the priority of the task k. In this study, however, the difference of priority among tasks is not considered, and w_k is set as 100 that is the

maximum score 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.

In Table 4.3, each UAV receives the information of pop-up tasks T6 and T7 from the MCC at time $t = 800$ sec., and therefore UAVs make an advertisement of themselves. Note that the waypoint number 100 denotes the RP. There are three disconnected sub-networks; i) group 1 consists of UAV1, UAV2, and UAV3, ii) group 2 consists of UAV4 and UAV5, and iii) group 3 consists of UAV6 and UAV7. At 801 sec., each sub-network tries to make a consensus on the PM. Note that the numbers of required iterations for each group are 2, 1, and 1, respectively (see Algorithm 4.10 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 4.11 line 3). At 803 sec., group 2 and 3 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, which 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 by 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 (see Algorithm 4.10 line 26).

Table 4.3 Detailed TA progress*

Time (sec.)	i	$K^{(i)}$	$\mathbf{A}^{(i)}$ $= [j,k,f]$	$\mathbf{L}_{app}^{(i)}$ $= [j,k,r]$	$\mathbf{p}^{(i)}$ $= [p_1,...]$	$\mathbf{t}^{(i)}$ $= [t_1,...]$
	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]
800	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]
	1	2	[3,6,60.599]			
	2	2	[3,6,60.599]			
	3	2	[3,6,60.599]			
801	4	2	[4,7,70.515]	unchanged	unchanged	unchanged
	5	2	[4,7,70.515]			
	6	2	[6,7,55.31]			
	7	2	[6,7,55.31]			
	1	2		[ ]		
	2	2		[ ]		
	3	2		[ ]		
802	4	3	unchanged	[ ]	unchanged	unchanged
	5	3		[5,7,1223]		
	6	3		[ ]		
	7	3		[7,7,1398]		
	1	3		[1,6,1463]		
	2	3		[2,6,1311]		
	3	3		[ ]		
803	4	3	unchanged	[5,7,1223]	unchanged	unchanged
	5	3		[5,7,1223]		
	6	3		[7,7,1398]		
	7	3		[7,7,1398]		

Time (sec.)	i	$K^{(i)}$	$\mathbf{A}^{(i)}$ $= [j, k, f]$	$\mathbf{L}_{app}^{(i)}$ $= [j, k, r]$	$\mathbf{p}^{(i)}$ $= [p_1, ...]$	$\mathbf{t}^{(i)}$ $= [t_1, ...]$
	1	3		[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]
804	4	4	unchanged	[ ]	[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]
	1	3	[3,6,60.599]			
	2	3	[3,6,60.599]			
	3	3	[3,6,60.599]			
805	4	1	[4,6,29.709]	unchanged	unchanged	unchanged
	5	1	[5,6,29.709]			
	6	1	[6,6,24.938]			
	7	1	[7,6,24.938]			
	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]
806	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]

* Variable definition: Table 4.1.

Monte Carlo Simulation

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 also 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 $y^{(i)}$ is set as 5. Among 10 tasks, the information regarding random two tasks, the other random two tasks, and the last random task is disseminated to agents at 800, 1,500, and 1,800 sec., respectively. On the other hand, the eight communication ranges are considered; 20, 30, 50, 100, 150, 200, 300, and 400 km. For each communication range, 100 random scenarios are applied. The performance of the proposed algorithms are compared with the performance of the GDAP algorithm [41], which is introduced in Section 4.4.

Figure 4.11 shows 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 provides 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 the dynamic environment, which will be discussed later in detail. As T_d becomes 0.2 sec., the scores of PCFA and TCFA are enhanced

significantly 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. 4.12, 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. This delay cancels out the advantage of TCFA. Therefore, when T_d is 0.2 sec., TCFA provides 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.

Figure 4.13 shows the average of mission completion time, which is defined as the time spent until every agent arrives 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. 4.11.

Figure 4.14 exhibits the averaged number of maximum communications with respect to communication ranges. The modified GDAP requires fewer communications because it does not include the consensus process. In PCFA and TCFA, more communications are required because more resets occur over shorter communication ranges. In connected networks, the number of communications by PCFA and TCFA are below 180, which is the upper bound calculated from Eq. (4.6).

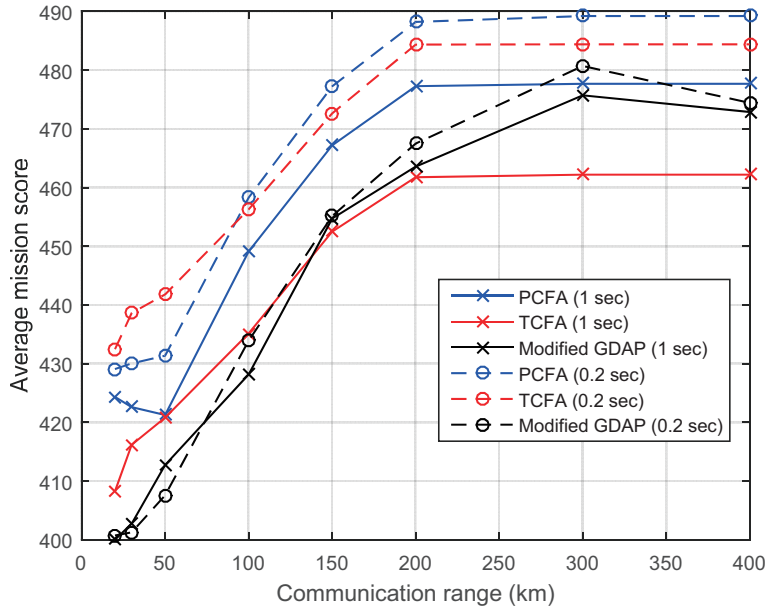


Figure 4.11 Average mission score

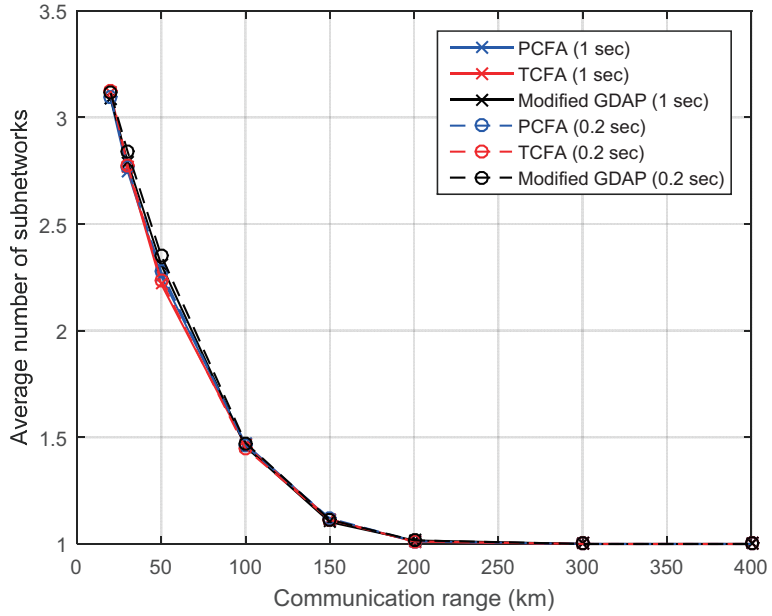


Figure 4.12 Average number of isolated sub-networks

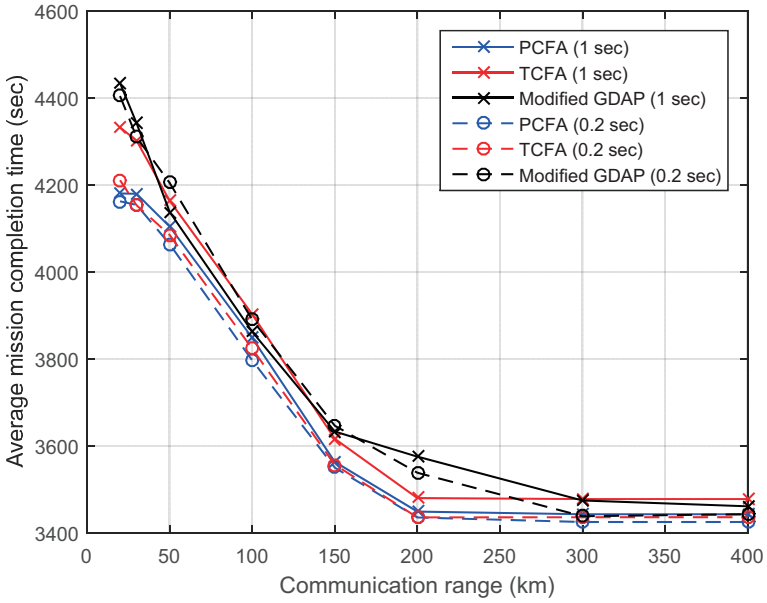


Figure 4.13 Average mission completion time

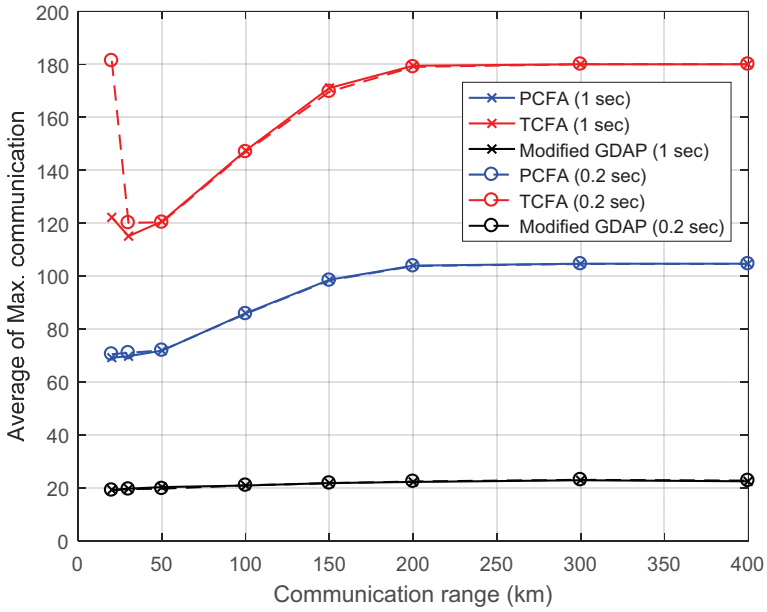


Figure 4.14 Average maximum communications

Figures 4.15∼4.17 show the statistical results of Monte Carlo simulations when T_d is 0.2 sec. On each box, the upper/lower 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. 4.16, outliers outside the whiskers are plotted together. Considering the outliers, the proposed algorithms sometimes use significantly more communications than the modified GDAP when the communication range is short; however, the trend is relaxed for the 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 using more communications.

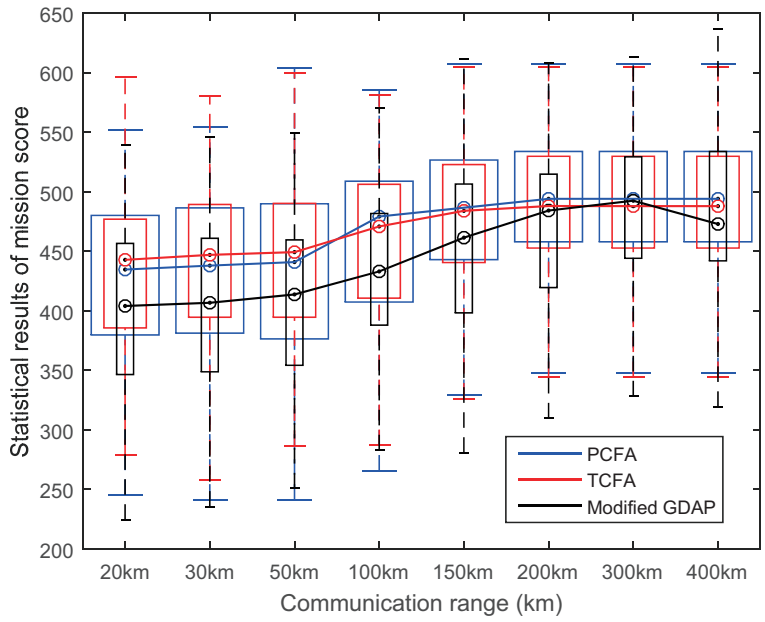


Figure 4.15 Statistical results of mission score

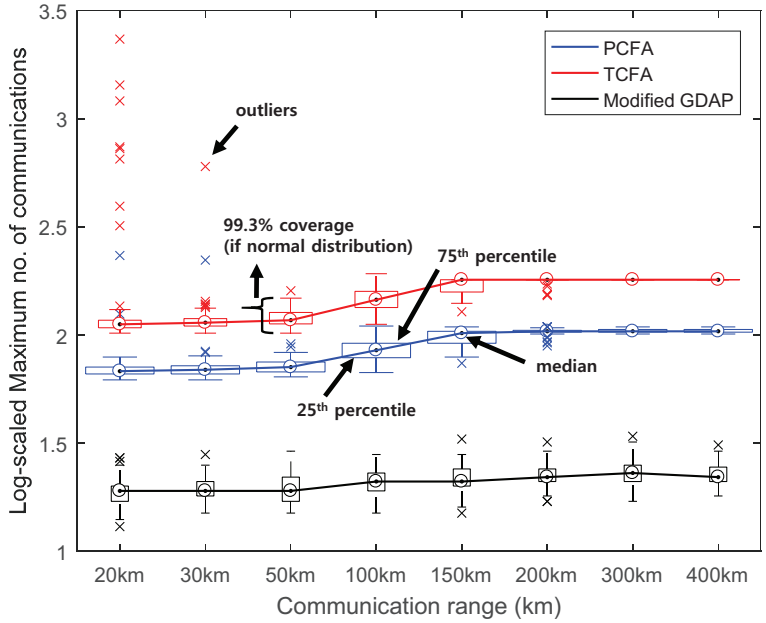


Figure 4.16 Statistical results of maximum communications (base=10)

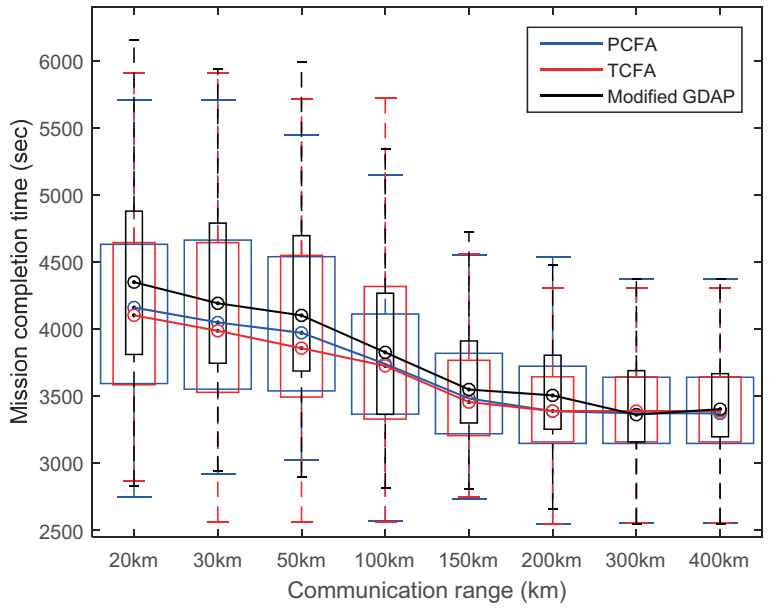


Figure 4.17 Statistical results of mission completion time

4.7.3 Discussion

Numerical simulation demonstrates that the proposed distributed coalition formation algorithms can be applied to the dynamic environment where time-varying as well as isolated sub-networks 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 suffer from several limitations. First, problem statement and 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. 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 including flight tests are required to verify the performance of the proposed algorithms. Finally, the scalability analysis with respect to the computation and communication in dynamic environment was not performed in this study. The time-varying network topology due to the limited communication range makes the convergence analysis very hard. Probability of isolated sub-networks makes this problem more challenging.

Chapter 5

Conclusions

5.1 Concluding Remarks

In this dissertation, the task allocation problem was studied, in which some tasks must be executed simultaneously by a predefined number of agents. The main results of this study are summarized in the following sections.

5.1.1 Problem Statement

The problem under consideration was systematically defined as an integer programming problem. In agent-based formulation, routing and overlapping coalition formation were explicitly treated by setting the agent visiting schedule as a decision variable. A directed acyclic graph constraint on a dependency graph was adopted for feasible routes of the cooperative timing mission. In task-based formulation, coalition members and task visiting order were considered as decision variables. As constraint equations could be satisfied relatively easily, heuristic methods based on task-based formulation performed better than those based on the agent-based formulation.

5.1.2 Centralized Task Allocation

If the mission control center frequently receives the required information on the mission environment and can broadcast the commands to the entire UAV fleet, then the centralized approach is recommended. In this dissertation, five

centralized methods to solve the task allocation problem of multiple UAVs for cooperative timing mission were presented: i) an exact algorithm, ii) agent-based sequential greedy algorithm (A-SGA), iii) task-based sequential greedy algorithm (T-SGA), iv) agent-based particle swarm optimization (A-PSO), and v) task-based particle swarm optimization (T-PSO). Numerical simulation results showed that the proposed methods successfully solved the given problems.

For dense mission environments, the average performance does not vary much by method. Thus, T-PSO is recommended for dense missions because it can provide a better or at least an equal solution compared to that of A-SGA within a relatively short time.

For relatively sparse missions having less than eight tasks, T-SGA is recommended because its performance corresponded to the solution of the exact algorithm. The computation time of T-SGA is less than the exact algorithm; however, T-SGA still suffers from the scalability issue as the number of tasks increases.

For relatively sparse missions having more than eight tasks, T-PSO is recommended. Compared to A-PSO, T-PSO was found to be a better solution than A-SGA, even while using a smaller runtime. Compared to T-SGA, the growth of T-PSO computation time is less sensitive to the number of tasks than to that of other methods.

5.1.3 Distributed Task Allocation

If the mission environment is dynamically changing, it is harder for the mission control center to receive real-time UAV information. In this case, the distributed approach is preferable, but a high level of agent autonomy is required.

In this dissertation, two market-based distributed task allocation algorithms were proposed: i) project manager-oriented coalition formation algorithm (PCFA)

and ii) task-oriented coalition formation algorithm (TCFA). Scalability analysis regarding time complexity and communication load was conducted in a connected network. Since the network can be disconnected during the mission in dynamic environment, proposed algorithms were extended to deal with the dynamic environment. For a comparative study, the greedy distributed allocation protocol (GDAP) was modified and implemented as a benchmark. A Monte Carlo simulation showed that the proposed algorithms performed better than the modified GDAP; however, additional communications are required.

For applications having strict limitation on communications and/or short communication range, the modified GDAP can be a reasonable choice for the task allocation. On the other hand, if the communication range is long enough to maintain the connected network, the PCFA or TCFA is recommended because they showed better performance within an upper bounded number of communications.

5.2 Future Research

Regarding the problem statement, the task allocation problem with more general temporal constraints, such as timed attacks and heterogeneous agents having various resources, is worth studying. The vehicle routing problem with multiple synchronization constraints (VRPMS) in [8] might be an appropriate starting point for this research. The SEAD mission consists of various tasks, including escort, electronic warfare (such as jamming), destruction of enemy radars or ground targets, and asset. In particular, some tasks may have temporal and/or spatial constraints. To formulate this kind of problem, the research conducted by Goel and Meisel [9] on electricity network maintenance, in which continuous variables were introduced, may prove helpful. In addition, Deng *et al.* [80] provided the graph theoretic scheme to investigate the violation of the task precedence. Alternatively, a relaxation of the target problem into linear programming or into a convex optimization problem may provide an upper bound of the objective function.

Regarding centralized task allocation, the approximation factor [81] of A-SGA should be analyzed. Since A-SGA is a deterministic and polynomial-time algorithm, the guaranteed performance might be favorable for practical application.

Regarding distributed task allocation, an asynchronous algorithm is desirable for real applications. Therefore, analyses on convergence and scalability should be performed for the asynchronous algorithm. The assumptions of network connectivity relying only on relative distance should be changed to reflect realistic network environments such as the log-distance path loss model. In addition, a sensitivity analysis on communication delay should be performed.

Note that the proposed algorithms were validated through Monte Carlo

numerical simulations in this study. Experimental demonstrations including ground and field flight tests are required for the verification of the proposed algorithms. For the implementation of the centralized approach, the communication capability of the ground station might be the key issue. For the implementation of the distributed approach, abrupt and irregular disconnection of the data link may degrade the efficiency of the task accomplishment, and therefore ground experiments with flight control system including data link should be performed prior to the field flight tests.

Bibliography

[1] Kraak, A. F., "F-35 Introduction: A Small Country Perspective," PowerPoint, Royal Netherlands Air Force, 2015. http://flytoazuresky.tistory.com/392.

[2] US Air Force, "Counterair Operations," Air Force Doctrine Document 2-1.1, 2008.

[3] Bolkcom, C., "Military Suppression of Enemy Air Defenses (SEAD): Assessing Future Needs," CRS Report for Congress RS21141, Congressional Research Service, Library of Congress, WA, USA, 2005.

[4] US DoD, "Unmanned Aircraft Systems Roadmap 2005-2030," 2005, pp. 1–62.

[5] McLain, T., "Coordinated Control of Unmanned Air Vehicles," Technical Report ASC-99-2426, Air Vehicles Directorate of the Air Force Research Laboratory, 1999.

[6] McLain, T. W., Chandler, P. R., Rasmussen, S., and Pachter, M., "Cooperative Control of UAV Rendezvous," *IEEE American Control Conference*, Arlington, VA, Jun. 2001.

[7] Xargay, E., Kaminer, I., Pascoal, A., Hovakimyan, N., Dobrokhodov, V., Cichella, V., Aguiar, A., and Ghabcheloo, R., "Time-Critical Cooperative Path Following of Multiple Unmanned Aerial Vehicles over Time-Varying

Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 499–516.

[8] Toth, P., and Vigo, D., *Vehicle Routing: Problems, Methods, and Applications*, 2nd Edition, Mathematical Optimization Society and the Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[9] Goel, A., and Meisel, F., "Workforce Routing and Scheduling for Electricity Network Maintenance with Downtime Minimization," *European Journal of Operational Research*, Vol. 231, No. 1, 2013, pp. 210–228.

[10] Olfati-Saber, R., and Murray, R. M., "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays," *IEEE Transactions on Automatic Control*, Vol. 49, No. 9, 2004, pp. 1520–1533.

[11] Ren, W., Beard, R. W., and Atkins, E. M., "Information Consensus in Multivehicle Cooperative Control," *IEEE Control Systems Magazine*, Vol. 27, No. 2, 2007, pp. 71–82.

[12] Bellingham, J., Tillerson, M., Richards, A., and How, J. P., "Multi-Task Allocation and Path Planning for Cooperating UAVs," *Conference on Cooperative Control and Optimization*, Gainsesville, FL, Nov. 2001.

[13] Alighanbari, M., *Task Assignment Algorithms for Teams of UAVs in Dynamic Environments*, MS Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge MA, Jun. 2004.

[14] Shaferman, V., and Shima, T., "Unmanned Aerial Vehicles Cooperative Tracking of Moving Ground Target in Urban Environments," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, 2008, pp. 1360–1371.

[15] Shima, T., and Schumacher, C., "Assigning Cooperating UAVs to Simultaneous Tasks on Consecutive Targets Using Genetic Algorithms," *Journal of the Operational Research Society*, Vol. 60, No. 7, 2009, pp. 973–982.

[16] Edison, E., and Shima, T., "Integrated Task Assignment and Path Optimization for Cooperating Uninhabited Aerial Vehicles Using Genetic Algorithms," *Computers and Operations Research*, Vol. 38, No. 1, 2011, pp. 340–356.

[17] Karaman, S., Shima, T., and Frazzoli, E., "A Process Algebra Genetic Algorithm," *IEEE Transactions on Evolutionary Computation*, Vol. 16, No. 4, 2012, pp. 489–503.

[18] Chandler, P. R., "Decentralized Control for an Autonomous Team," *AIAA 2nd Unmanned Unlimited Conference*, San Diego, CA, Sep. 2003.

[19] Alighanbari, M., *Robust and Decentralized Task Assignment Algorithms for UAVs*, Ph.D. Dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge MA, Sep. 2007.

[20] Choi, H., Kim, Y., and Kim, H., "Genetic Algorithm Based Decentralized Task Assignment for Multiple UAVs in Dynamic Environments," *International Journal of Aeronautical and Space Sciences*, Vol. 12, No. 2, 2011, pp. 163–174.

[21] Shaferman, V., and Shima, T., "Task Assignment and Motion Planning for Multiple UAVs Tracking Multiple Targets in Urban Environments," *AIAA Guidance, Navigation, and Control Conference*, Chicago, IL, Aug. 2009.

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

[23] Ponda, S., Redding, J., Choi, H.-L., How, J. P., Vavrina, M., and Vian, J., "Decentralized Planning for Complex Missions with Dynamic Communication Constraints," *IEEE American Control Conference*, Baltimore, MD, Jul. 2010.

[24] Johnson, L., Choi, H.-L., and How, J. P., "Hybrid Information and Plan Consensus in Distributed Task Allocation," *AIAA Guidance, Navigation, and Control Conference*, Boston, MA, Aug. 2013.

[25] Gerkey, B. P., and Matarić, M. J., "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *The International Journal of Robotics Research*, Vol. 23, No. 9, 2004, pp. 939–954.

[26] Sandholm, T. W., and Lesser, V. R., "Coalition Formation among Bounded Rational Agents," *International Joint Conference on Artificial Intelligence*, Quebec, Canada, Aug. 1995.

[27] Ketchpel, S., "Forming Coalitions in the Face of Uncertain Rewards," *Twelfth National Conference on Artificial Intelligence*, Seattle, WA, Jul. 1994.

[28] Zlotkin, G., and Rosenschein, J. S., "Coalition, Cryptography, and Stability: Mechanisms for Coalition Formation in Task Oriented Domains," *National Conference on Artificial Intelligence*, Seattle, WA, Jul. 1994.

[29] Sandhlom, T. W., and Lesser, V. R., "Coalitions Among Computationally Bounded Agents," *Artificial Intelligence*, Vol. 94, No. 1, 1997, pp. 99–137.

[30] Sandholm, T., Larson, K., Andersson, M., Shehory, O., and Tohmé, F., "Coalition Structure Generation with Worst Case Guarantees," *Artificial Intelligence*, Vol. 111, No. 1, 1999, pp. 209–238.

[31] Rahwan, T., Ramchurn, S. D., Jennings, N. R., and Giovannucci, A., "An Anytime Algorithm for Optimal Coalition Structure Generation," *Journal of Artificial Intelligence Research*, Vol. 34, No. 1, 2009, pp. 521–567.

[32] Chalkiadakis, G., and Boutilier, C., "Sequentially Optimal Repeated Coalition Formation under Uncertainty," *Autonomous Agents and Multi-Agent Systems*, Vol. 24, No. 3, 2012, pp. 441–484.

[33] Rahwan, T., Michalak, T. P., Elkind, E., Faliszewski, P., Sroka, J., Wooldridge, M., and Jennings, N. R., "Constrained Coalition Formation." *AAAI Conference on Artificial Intelligence*, San Francisco, CA, Aug. 2011.

[34] Wang, W., and Jiang, Y., "Community-Aware Task Allocation for Social Networked Multiagent Systems," *IEEE Transactions on Cybernetics*, Vol. 44, No. 9, 2014, pp. 1529–1543.

[35] Liang, X., and Xiao, Y., "Studying Bio-Inspired Coalition Formation of Robots for Detecting Intrusions Using Game Theory," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 40, No. 3, 2010, pp. 683–693.

[36] Haque, M., Egerstedt, M., and Rahmani, A., "Multilevel Coalition Formation Strategy for Suppression of Enemy Air Defenses Missions," *Journal of Aerospace Information Systems*, Vol. 10, No. 6, 2013, pp. 287–296.

[37] Dang, V. D., Dash, R. K., Rogers, A., and Jennings, N. R., "Overlapping Coalition Formation for Efficient Data Fusion in Multi-Sensor Networks," *National Conference on Artificial Intelligence*, Boston, MA, Jul. 2006.

[38] Chalkiadakis, G., Elkind, E., Markakis, E., Polukarov, M., and Jennings, N. R., "Cooperative Games with Overlapping Coalitions," *Journal of Artificial Intelligence Research*, Vol. 39, No. 1, 2010, pp. 179–216.

[39] Zick, Y., Chalkiadakis, G., and Elkind, E., "Overlapping Coalition Formation Games: Charting the Tractability Frontier," *International Conference on Autonomous Agents and Multiagent Systems*, Valencia, Spain, Jun. 2012.

[40] Shehory, O., and Kraus, S., "Methods for Task Allocation via Agent Coalition Formation," *Artificial Intelligence*, Vol. 101, No. 1, 1998, pp. 165–200.

[41] Weerdt, M. d., Zhang, Y., and Klos, T., "Multiagent Task Allocation in Social Networks," *Autonomous Agents and Multi-Agent Systems*, Vol. 25, No. 1, 2012, pp. 46–86.

[42] Oh, G., Kim, Y., Ahn, J., and Choi, H.-L., "Market-Based Task Assignment for Cooperative Timing Missions over Networks with Limited Connectivity," *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, FL, Jan. 2015.

[43] Whitten, A. K., Choi, H.-L., Johnson, L. B., and How, J. P., "Decentralized Task Allocation with Coupled Constraints in Complex Missions," *IEEE American Control Conference*, San Francisco, CA, Jun. 2011.

[44] Ramchurn, S. D., Polukarov, M., Farinelli, A., Truong, C., and Jennings, N. R., "Coalition Formation with Spatial and Temporal Constraints," *In-*

ternational Conference on Autonomous Agents and Multiagent Systems (AAMAS-10), Toronto, Canada, May 2010.

[45] Sujit, P., George, J., and Beard, R., "Multiple UAV Task Allocation Using Particle Swarm Optimization," *AIAA Guidance, Navigation, and Control Conference*, Honolulu, HI, Aug. 2008.

[46] Manathara, J. G., Sujit, P., and Beard, R. W., "Multiple UAV Coalitions for a Search and Prosecute Mission," *Journal of Intelligent and Robotic Systems*, Vol. 62, No. 1, 2011, pp. 125–158.

[47] Oh, G., Kim, Y., Ahn, J., and Choi, H.-L., "PSO-Based Optimal Task Allocation for Cooperative Timing Missions," *20th IFAC Symposium on Automatic Control in Aerospace*, Sherbrooke, Canada, Aug. 2016.

[48] Smith, R. G., "The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computers*, Vol. 29, No. 12, 1980, pp. 1104–1113.

[49] Farber, D. J., and Larson, K. C., "The Structure of a Distributed Computing System-Software," *Symposium on Computer-Communications Networks and Teletraffic*, New York, NY, Apr. 1972.

[50] Dias, M. B., Zlot, R., Kalra, N., and Stentz, A., "Market-based Multirobot Coordination: A Survey and Analysis," *Proceedings of the IEEE*, Vol. 94, No. 7, 2006, pp. 1257–1270.

[51] Bertsekas, D. P., "The Auction Algorithm for Assignment and Other Network Flow Problems," Technical Report LIDS-P-1908, MIT, Cambridge, MA, Sep. 1989.

[52] Gerkey, B. P., and Matari, M. J., "Sold!: Auction Methods for Multirobot Coordination," *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, 2002, pp. 758–768.

[53] Vig, L., and Adams, J. A., "Coalition Formation: From Software Agents to Robots," *Journal of Intelligent and Robotic Systems*, Vol. 50, No. 1, 2007, pp. 85–118.

[54] Maza, I., Kondak, K., Bernard, M., and Ollero, A., "Multi-UAV Cooperation and Control for Load Transportation and Deployment," *Journal of Intelligent and Robotic Systems*, Vol. 57, No. 1-4, 2010, pp. 417–449.

[55] Khamis, A. M., Elmogy, A. M., and Karray, F. O., "Complex Task Allocation in Mobile Surveillance Systems," *Journal of Intelligent and Robotic Systems*, Vol. 64, No. 1, 2011, pp. 33–55.

[56] Sujit, P., George, J., and Beard, R., "Multiple UAV Coalition Formation," *IEEE American Control Conference*, Seattle, WA, Jun. 2008.

[57] Service, T. C., and Adams, J. A., "Coalition Formation for Task Allocation: Theory and Algorithms," *Autonomous Agents and Multi-Agent Systems*, Vol. 22, No. 2, 2011, pp. 225–248.

[58] Das, G., McGinnity, T., and Coleman, S., "Simultaneous Allocations of Multiple Tightly-Coupled Multi-Robot Tasks to Coalitions of Heterogeneous Robots," *IEEE International Conference on Robotics and Biomimetics*, Bali, Indonesia, Dec. 2014.

[59] Das, G. P., McGinnity, T. M., Coleman, S. A., and Behera, L., "A Distributed Task Allocation Algorithm for a Multi-Robot System in Health-

care Facilities," *Journal of Intelligent and Robotic Systems*, Vol. 80, No. 1, 2015, pp. 33–58.

[60] Beard, R. W., and McLain, T. W., "Multiple UAV Cooperative Search under Collision Avoidance and Limited Range Communication Constraints," *IEEE Conference on Decision and Control*, Maui, HI, Dec. 2003.

[61] Sujit, P., and Beard, R., "Distributed Sequential Auctions for Multiple UAV Task Allocation," *IEEE American Control Conference*, New York, NY, Aug. 2007.

[62] Ponda, S. S., Johnson, L. B., Kopeikin, A. N., Choi, H.-L., and How, J. P., "Distributed Planning Strategies to Ensure Network Connectivity for Dynamic Heterogeneous Teams," *IEEE Journal on Selected Areas in Communications*, Vol. 30, No. 5, 2012, pp. 861–869.

[63] Pujol-Gonzalez, M., Cerquides, J., Meseguer, P., Rodríguez-Aguilar, J., and Tambe, M., "Engineering the Decentralized Coordination of UAVs with Limited Communication Range," *Advances in Artificial Intelligence*, Vol. 8109, *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2013, pp. 199–208.

[64] George, J., Sujit, P., and Sousa, J., "Coalition Formation with Communication Delays and Maneuvering Targets," *AIAA Guidance, Navigation, and Control Conference*, Toronto, Canada, Aug. 2010.

[65] Sujit, P., Manathara, J., Ghose, D., and de Sousa, J., "Decentralized Multi-UAV Coalition Formation with Limited Communication Ranges," *Handbook of Unmanned Aerial Vehicles*, Springer, Berlin, Germany, 2014, pp. 2021–2048.

[66] Balmas, F., "Displaying Dependence Graphs: A Hierarchical Approach," *Journal of Software Maintenance and Evolution: Research and Practice*, Vol. 16, No. 3, 2004, pp. 151–185.

[67] Gross, J. L., and Yellen, J., *Handbook of Graph Theory*, CRC press, Boca Raton, FL, 2003.

[68] Richards, A., Bellingham, J., Tillerson, M., and How, J., "Coordination and Control of Multiple UAVs," *AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, Aug. 2002.

[69] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms*, MIT University Press, Cambridge, MA, 2001.

[70] Kenndy, J., and Eberhart, R., "Particle Swarm Optimization," *IEEE International Conference on Neural Networks*, Perth, Australia, Nov. 1995.

[71] Banks, A., Vincent, J., and Anyakoha, C., "A Review of Particle Swarm Optimization. Part I: Background and Development," *Natural Computing*, Vol. 6, No. 4, 2007, pp. 467–484.

[72] Banks, A., Vincent, J., and Anyakoha, C., "A Review of Particle Swarm Optimization. Part II: Hybridisation, Combinatorial, Multicriteria and Constrained Optimization, and Indicative Applications," *Natural Computing*, Vol. 7, No. 1, 2008, pp. 109–124.

[73] Clerc, M., and Kennedy, J., "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 1, 2002, pp. 58–73.

[74] Laskari, E. C., Parsopoulos, K. E., and Vrahatis, M. N., "Particle Swarm Optimization for Integer Programming," *World Congress on Computational Intelligence*, Honolulu, HI, May 2002.

[75] Kennedy, J., "Particle Swarm Optimization," *Encyclopedia of Machine Learning*, edited by Sammut, C., and Webb, G.I., Springer, 2011, pp. 760–766.

[76] Vela, A. E., Solak, S., Clarke, J.-P. B., Singhose, W. E., Barnes, E. R., and Johnson, E. L., "Near Real-Time Fuel-Optimal En Route Conflict Resolution," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 11, No. 4, 2010, pp. 826–837.

[77] Devadoss, S. L., and O'Rourke, J., *Discrete and Computational Geometry*, Princeton University Press, Princeton, NJ, 2011.

[78] Arslan, G., Marden, J. R., and Shamma, J. S., "Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 129, No. 5, 2007, pp. 584–596.

[79] Rawlings, J. O., Pantula, S. G., and Dickey, D. A., *Applied Regression Analysis: A Research Tool*, Springer, New York, NY, 1998.

[80] Deng, Q., Yu, J., and Mei, Y., "Deadlock-Free Consecutive Task Assignment of Multiple Heterogeneous Unmanned Aerial Vehicles," *Journal of Aircraft*, Vol. 51, No. 2, 2014, pp. 596–605.

[81] Williamson, D. P., and Shmoys, D. B., *The Design of Approximation Algorithms*, Cambridge University Press, Cambridge, England, 2011.

Appendix

Directed Acyclic Graph Constraint on Dependency Graph

Task allocation (TA) algorithm for cooperative timing missions should provide the *proper* visiting schedules so that the tasks can be performed simultaneously. Consider a following case that involves three agents and four tasks; $z_1 = z_2 = z_3 = z_4 = 2$, $y^{(1)} = y^{(2)} = y^{(3)} = 3$, and

$$\mathbf{p}^{(1)} = [4\ 3\ 2\ 0],\ \mathbf{p}^{(2)} = [4\ 1\ 3\ 0],\ \mathbf{p}^{(3)} = [2\ 1\ 0\ 0] \tag{A.1}$$

The above path vectors satisfy the constraints of Eqs. (2.2) and (2.3), but the task visiting order is twisted. According to the path vectors of agents 1 and 2, the task visiting order is $4 \to 1 \to 3 \to 2$, whereas the order is $4 \to 3 \to 2 \to 1$ in view of agents 1 and 3. Due to the conflict of the visiting order between tasks, it is not possible to perform simultaneous arrival.

Let us consider another example, where $\mathbf{p}_3$ is modified as follow to resolve the conflict, which allows simultaneous arrivals.

$$\mathbf{p}^{(1)} = [4\ 3\ 2\ 0],\ \mathbf{p}^{(2)} = [4\ 1\ 3\ 0],\ \mathbf{p}^{(3)} = [1\ 2\ 0\ 0] \tag{A.2}$$

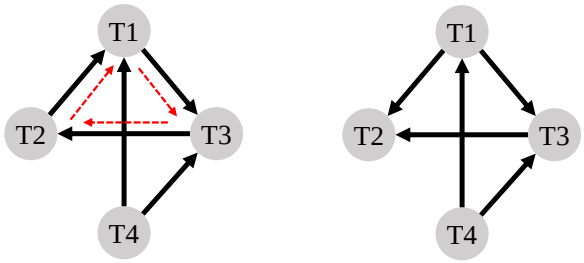
Therefore, to guarantee simultaneous involvements for all given tasks, the path matrix should be constrained in the problem statement. For the generalized expression of the constraint on the task visiting order, let us introduce a dependency graph $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$, where a directed edge-set $\mathcal{E}(\mathbf{P})$ is defined as in

Eq. (2.5). If a directed cycle exists in the graph $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$, then the path matrix cannot allow simultaneous arrivals, as shown in Proposition A.1.

Proposition A.1. *Let us consider a TA problem for a cooperative timing mission as defined in Eqs.* (2.1)–(2.4)*. If there exists a directed cycle in the dependency graph* $G = (\mathcal{K}, \mathcal{E}(\boldsymbol{P}))$*, then the path matrix* $\boldsymbol{P}$ *cannot allow simultaneous arrivals.*

Proof. Suppose that a directed cycle exists in the dependency graph $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$ and the path matrix $\mathbf{P}$ can allow simultaneous arrivals.
It can be stated that the directed cycle consists of arbitrary $m \in \{2, 3, ..., M\}$ tasks among M tasks. For convenience, let us assign these tasks as task 1, task 2,..., and task m. Then, there exist m precedents among the tasks, i.e., $1 \to 2$, $2 \to 3$,..., $m-1 \to m$, $m \to 1$. The corresponding arrival time for each task may be uniquely determined as t_1, t_2,..., t_m. According to the aforementioned precedents, it can be stated that $t_1 < t_2$, $t_2 < t_3$,..., $t_{m-1} < t_m$, which yields $t_1 < t_m$. The last precedence $m \to 1$ yields an inequality $t_m < t_1$, which leads to a contradiction. Therefore, the path matrix $\mathbf{P}$ cannot allow simultaneous arrivals when its dependency graph $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$ has any directed cycles. □

For the cases of Eqs. (A.1) and (A.2), the corresponding dependency graphs can be graphically illustrated as in Fig. A.1. Note that a directed cycle exists in the case of (A.1).



(a) **Example 1** (b) **Example 2**

Figure A.1 Graphical representation of the dependency graph $G = (\mathcal{K}, \mathcal{E}(\mathbf{P}))$.

국문초록

무인항공기의 자율비행 기술이 성숙함에 따라 무인항공기에 요구되는 임무의 복잡도와 정밀도가 증가하고 있다. 최근에는 단일 무인항공기에 의한 감시정찰 임무에서 나아가 다수의 무인항공기의 협력적인 임무수행 능력에 관한 연구가 활발히 수행되고 있다. 본 연구에서는 무인항공기의 협업에 의한 잠재력을 최대한으로 활용하기 위하여 다수의 무인항공기가 동시에 수행해야 하는 임무를 고려하였다. 이러한 임무로는 위험도가 높은 방어 시스템을 동시에 공격하는 임무, 넓은 재난 지역을 다수의 무인기가 동시에 수색, 물품지원, 구조 등을 수행하는 임무, 그리고 무거운 물체를 다수의 무인항공기가 협력하여 수송하는 임무 등을 고려할 수 있다. 이와 같이 복잡한 임무를 관리하기 위해 지상 조종사는 다수의 무인항공기를 관제하여야 하며, 이 과정에서 과도한 업무부하는 조종사 실수를 유발하여 임무수행 효율저하로 이어질 수 있다.

본 연구에서는 다수 무인항공기의 동시도달을 고려한 협력 임무할당 문제를 정수계획법으로 정식화하고, 중앙집중형 임무할당 방식과 분산형 임무할당 방식을 연구하였다. 무인항공기로부터 수집된 정보를 기반으로 최적에 가까운 임무할당을 결정하는 중앙집중형 임무할당 방식으로는 모든 해 공간을 탐색하여 최적해를 계산하는 방식, 경험적인 법칙을 통해 신속하게 해를 결정하는 방식, 그리고 메타휴리스틱 기법의 일종인 군집 최적화 기법을 활용하는 방식을 제안하였다. 분산형 임무할당 방식으로는 개별 무인항공기는 모든 무인항공기가 아닌 이웃 무인항공기들과만 정보를 교류하고, 이를 통하여 자율적으로 임무를 할당하는 기법을

제안하였다. 제한된 통신반경에 따른 실시간 네트워크 위상변화 상황을 고려하기 위하여 집결지 개념을 도입하였으며, 연결된 네트워크 상황에 대하여 수렴성과 확장성을 분석하였다. 제안한 기법들의 성능을 검증하기 위하여 적 대공망 제압작전 시나리오에 대한 수치 시뮬레이션을 수행하고, 제안한 기법 간의 성능을 비교 분석하였다.

주요어: 임무 할당, 무인항공기, 동시도달, 중앙집중형 임무할당, 분산형 임무할당
학번: 2010-20693