


# An Integrated Decision-making Framework of a Heterogeneous Aerial Robotic Swarm for Cooperative Tasks with Minimum Requirements

Journal Title  
XX(X):1-??  
©The Author(s) 2018  
Reprints and permission:  
sagepub.co.uk/journalsPermissions.nav  
DOI: 10.1177/ToBeAssigned  
www.sagepub.com/  


Inmo Jang<sup>1</sup>, Hyo-Sang Shin<sup>1</sup>, Antonios Tsourdos<sup>1</sup>, Junho Jeong<sup>2</sup>, Seungkeun Kim<sup>2</sup>, and Jinyoung Suk<sup>2</sup>

## Abstract

Given a cooperative mission consisting of multiple tasks spatially distributed, an aerial robotic swarm's decision-making issues include team formation, team-to-task assignment, agent-to-work-position assignment, and trajectory optimisation with collision avoidance. The problem becomes even more complicated when involving heterogeneous agents, tasks' minimum requirements, and fair allocation. This paper formulates all the combined issues as an optimisation problem and then proposes an integrated framework that addresses the problem in a decentralised fashion. We approximate and decouple the complex original problem into three subproblems (i.e., coalition formation, position allocation, and path planning), which are sequentially addressed by three different proposed modules. The coalition formation module based on game theories deals with a max-min problem, the objective of which is to partition the agents into disjoint task-specific teams in a way that balances the agents' work resources in proportion to the task's minimum workload requirements. For agents assigned to the same task, given reasonable assumptions, the position allocation subproblem can be efficiently addressed in terms of computational complexity. For the trajectory optimisation, we utilise an MPC-SCP (Model Predictive Control and Sequential Convex Programming) algorithm, which reduces the size of the problem so that the agents can generate collision-free trajectories in a real-time basis. As a proof of concept, we implement the framework into a UAV swarm's cooperative stand-in jamming mission scenario and show its feasibility, fault tolerance, and near-optimality based on numerical experiment.

## Keywords

Unmanned aerial vehicles, Swarm robotics, Mission planning, Task allocation, Coalition formation, Path planning.

## 1 Introduction

A networked system of a large number of aerial robots, called (*aerial*) *robotic swarm* or *UAV swarm*, has been attracting many researchers' interest because of its promising advantages such as its versatility, robustness, and adaptiveness<sup>1-3</sup>. To name a few, possible applications of UAV swarms include environmental monitoring<sup>4</sup>, ad-hoc network relay<sup>5</sup>, disaster management<sup>6</sup>, and cooperative military missions<sup>7</sup>. However, there still exist various technical challenges to realise a robotic swarm into real-life applications, for examples in view of autonomous decision-making domain, multi-robot task allocation problem<sup>8-10</sup> and trajectory optimisation including collision avoidance<sup>11,12</sup>.

Cooperation is essential for a robotic swarm: robots (or agents) with cheaper components can be realised through

mass production in lower cost, but each of them eventually has limited capability to complete a single task alone<sup>13</sup>. Given multiple tasks spatially distributed, the robots have to partition themselves into disjoint task-specific robot teams (or coalitions). This decision-making issue is referred to as *coalition formation* problem<sup>14</sup> or *ST-MR multi-robot task allocation* problem<sup>8</sup>. Even though the robotic swarm was identically manufactured, the agents may be

<sup>1</sup>Centre for Autonomous and Cyber-Physical Systems, Cranfield University, UK

<sup>2</sup>Chungnam National University, Republic of Korea

### Corresponding author:

Hyo-Sang Shin, Centre for Autonomous and Cyber-Physical Systems, Cranfield University, Cranfield, Bedford, MK43 0TH, UK.

Email: h.shin@cranfield.ac.uk

heterogeneous in terms of the currently-available work resources because, for examples, it is probably not possible to charge every robot's battery fully before a mission due to the large cardinality. Such heterogeneity makes the coalition formation problem more complicated. Moreover, the tasks may have their minimum workload requirements, which have to be cooperatively fulfilled by the agents' resources. Having extra resources, in some cases, it is desirable to equitably distribute them in proportion to the minimum requirements to improve the system's fault tolerance.

Each task-specific team of robots also have to make decisions regarding which (spatial) work positions to go specifically, called *position allocation* problem. Besides, the agents' trajectories towards the assigned positions should be optimised, guaranteeing that any collision with other agents or obstacles will not happen during the mission. What makes the entire problem more involved is that the trajectories affect the expected working resources of the agents when they arrive at and execute their assigned tasks. Therefore, such effects should also be considered in the coalition formation and position allocation problems, and thereby the combined decision-making problem becomes much more complicated than either of individual problems.

Over the last few years, some researchers have begun to address problems of simultaneously finding robot-task assignments and trajectories. One typical strategy is to regard such a complex decision-making issue as a single optimisation problem (e.g., minimising the sum of distance travelled by all the agents towards their assigned positions) and solve it either in a centralised<sup>15</sup> or decentralised manner<sup>16</sup>. In particular, Turpin et al.<sup>16</sup> show that assuming homogeneous robots the optimal assignment minimising velocity-squared trajectories without considering collision avoidance can be actually collision-free and that a suboptimal outcome can be obtained in a decentralised manner. However, one drawback of this work is that a robot's trajectory is assumed as a straight line, which may fail to deal with obstacle avoidance. Alternatively, there have also been decoupling strategies, where tasks are first assigned and then trajectories are planned<sup>17–19</sup>. This type of approaches is advantageous in the sense that each subproblem can be substituted as required depending on practical applications considered. The existing works<sup>18,19</sup> utilise for task allocation a stochastic-policy-based method and an auction-based method, respectively, and exploit an MPC-SCP (Model Predictive Control and Sequential Convex Programming) algorithm for path planning<sup>20</sup>, which was experimentally shown to be implementable for a multi-robot system on a real-time basis as well as has the potential to

consider obstacle avoidance. However, all of the existing studies reviewed consider neither heterogeneous robots nor fair allocation concerning task requirements, which will be addressed in this paper.

We first formulate all the prescribed decision-making issues of a heterogeneous robotic swarm as a combined optimisation problem, and then propose an integrated framework that addresses the problem in a decentralised fashion. Our approach approximates and decouples the problem into three subproblems, i.e., coalition formation, position allocation, and path planning, which are sequentially addressed by three different subroutine algorithms. Due to the large cardinality of a robotic swarm system, directly obtaining robot-to-position assignment can be restrictive regarding computational complexity. To avoid this hinderance, the proposed approach firstly partitions the robots into disjoint task-specific coalitions, followed by the position allocation subproblem for the robots assigned to the same task. For the coalition formation subproblem, we propose a game-theoretical method in which each agent unilaterally selects a task-specific coalition with consideration of fair allocation regarding the given tasks' requirements. We show that, given reasonable assumptions, the position allocation subproblem can be efficiently addressed in terms of computational complexity even using a simple sorting algorithm. For the trajectory optimisation, we utilise an MPC-SCP algorithm because of its advantages (e.g., real-time implementability). As a proof of concept, we implement the framework into a cooperative stand-in jamming mission scenario using a swarm of micro UAVs and show its feasibility, fault tolerance, and near-optimality based on numerical experiment.

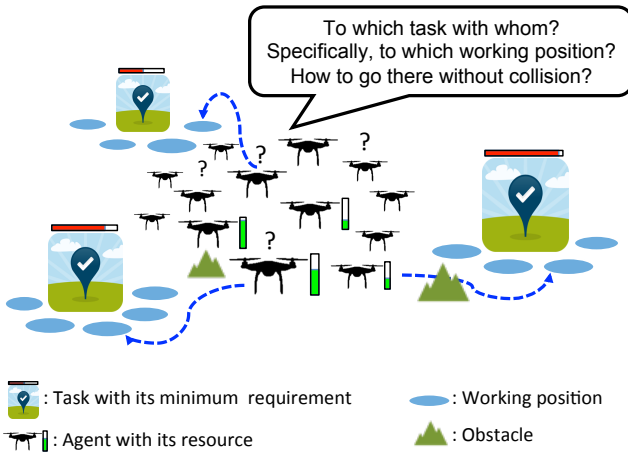
This paper is organised as follows. Section 2 defines the original complex problem, which is decoupled into the three subproblems in Section 3. Then, Sections 4, 5, and 6 propose the subroutines for the coalition formation, the position allocation, and the path planning subproblems, respectively. In Section 7, we propose the integrated framework consisting of the three subroutines and discuss its properties. Section 8 shows the results of numerical experiments using the framework on a UAV swarm's cooperative stand-in jamming mission.

## 2 Problem statement: the original problem

We have a set of *agents*  $\mathcal{A} = \{a_1, a_2, \dots, a_{n_a}\}$ , and a set of a fewer number of spatially distributed *tasks*  $\mathcal{T} = \{t_1, t_2, \dots, t_{n_t}\}$ , where  $n_t < n_a$ , to be collaboratively executed by the agents. Each task  $t_j$  has a set of (spatial) *work positions*  $\mathcal{P}_j = \{p_{m_1}, p_{m_2}, \dots\}$ , where every

**Table 1.** Nomenclature

Symbol	Description
$\mathcal{T}$	A set of $n_t$ tasks $\{t_1, t_2, \dots, t_{n_t}\}$ ;
$\mathcal{A}$	A set of $n_a$ agents $\{a_1, a_2, \dots, a_{n_a}\}$ ;
$\mathcal{S}_j$	The task-specific coalition of agents for task $t_j$ ;
$\Pi$	The set of coalitions $\Pi = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_{n_t}, \mathcal{S}_0\}$ ;
$\mathcal{O}$	A set of $n_o$ obstacles $\{o_1, o_2, \dots, o_{n_o}\}$ ;
$\mathcal{P}_j$	A set of working positions $\{p_{m_1}, p_{m_2}, \dots\}$ where agents can execute task $t_j$ ;
$\mathcal{P}$	The set of the entire working positions i.e., $\mathcal{P} := \cup_{\forall t_j \in \mathcal{T}} \mathcal{P}_j$ ;
$w_{im}$	Agent $a_i$ 's (expected) work resource at position $p_m$ ;
$c_{im}$	The cost if agent $a_i$ works at position $p_m$ ;
$\bar{w}_{ij}$	Agent $a_i$ 's abstracted (expected) work resource for task $t_j$ ;
$\bar{c}_{ij}$	The abstracted cost of agent $a_i$ for task $t_j$ ;
$w_{i,0}$	Agent $a_i$ 's initial work resource;
$p_{j,0}$	task $t_j$ 's central position;
$R_j$	The minimum requirement for task $t_j$ to be completed;
$\alpha_j$	RSI (Requirement Satisfaction Index) for task $t_j$ (Eqn. (3));
$\mathbf{x}_i$	The position and velocity vector of agent $a_i$ ;

**Figure 1.** A brief illustration of the decision-making issues considered in this paper

$p_m \in \mathbb{R}^{n_d \times 1}$  is an  $n_d$ -dimensional Euclidean-space position vector at which an agent can execute the task. Each work position is only associated with a single task, i.e.,  $\mathcal{P}_{j_1} \cap \mathcal{P}_{j_2} = \emptyset$  for  $\forall j_1 \neq j_2$ . Without loss of generality, let  $\mathcal{P}_1 = \{p_1, \dots, p_{|\mathcal{P}_1|}\}$ ,  $\mathcal{P}_2 = \{p_{|\mathcal{P}_1|+1}, \dots, p_{|\mathcal{P}_1|+|\mathcal{P}_2|}\}, \dots, \mathcal{P}_{n_t} = \{p_{|\mathcal{P}_{n_t-1}|+1}, \dots, p_{|\mathcal{P}_{n_t-1}|+|\mathcal{P}_{n_t}|}\}$ . We denote the set of the entire work positions by  $\mathcal{P} := \cup_{\forall t_j \in \mathcal{T}} \mathcal{P}_j$ . Furthermore, we have a set of (static) *obstacles*  $\mathcal{O} = \{o_1, o_2, \dots, o_{n_o}\}$  the agents have to avoid during the mission, where each  $o_q \in \mathbb{R}^{n_d \times 1}$  indicates the central position vector of the  $q$ -th obstacle.

Each agent  $a_i$  has its (expected available) *work resource* (or *work capacity*)  $w_{im}$ , which varies depending on work position  $p_m \in \mathcal{P}$ . For examples, given the agent's initial work resource (denoted by  $w_{i,0}$ ), the cost to transition to work position  $p_m$  (denoted by  $c_{im}$ ) may result in a different level of expected work capacity at last. This is also the case if the work position affects the agent's work efficiency (e.g., in the cooperative jamming mission<sup>7</sup>, a position closer to a target radar provides higher jamming effectiveness). Considering this, work capacity  $w_{im}$  can be defined as

$$w_{im} := \eta_m(w_{i,0} - c_{im}), \quad (1)$$

where  $\eta_m \in [0, 1]$  is the *work efficiency ratio* associated with position  $p_m$ .

For each task  $t_j$  to be completed, its *minimum (workload) requirement*  $R_j$  should be met by its (*task-specific*) *coalition* of the agents, denoted by  $\mathcal{S}_j \subseteq \mathcal{A}$  (note that  $\mathcal{S}_{j_1} \cap \mathcal{S}_{j_2} = \emptyset$  if  $j_1 \neq j_2$ ). We regard that the task is completed if the aggregated work capacities of the coalition exceed the minimum requirement. Having extra agents besides those for satisfying all the tasks' minimum requirements, it would be desirable to equitably distribute them as proportional to the requirements as possible to improve the system robustness.

Furthermore, there is a particular final time  $t_f$  by when all the agents should reach appropriate work positions without any collisions. Such a deadline may be an important factor for success in urgent missions, for instances, those in military applications or those in the search and rescue domain.

In such a prescribed mission environment, the robotic swarm's main decision-making issues may include:

1. To form task-specific coalitions and choose work positions in a way that (a) satisfies every task's minimum requirement at least and (b) maximises the agents' work capacities by reducing travelling costs as well as exploiting positions with higher work efficiency;
2. To equitably distribute the agents' work capacities in proportional to the tasks' requirements as possible; and
3. To generate shortest trajectories towards the selected positions, while avoiding any collisions with other agents or obstacles.

This problem as a whole is briefly illustrated in Figure 1 and can be formally defined as follows:

**Problem 1.** (Original problem).

$$\max_{\{y_{im}\}} \left( \min_{\forall t_j \in \mathcal{T}} \alpha_j \right), \quad (2)$$

where

$$\alpha_j := \frac{\sum_{\forall a_i \in \mathcal{S}_j} \sum_{\forall p_m \in \mathcal{P}_j} w_{im} y_{im}}{R_j}, \quad (3)$$

subject to

$$\alpha_j \geq 1, \quad \forall t_j \in \mathcal{T}, \quad (4)$$

$$\sum_{\forall p_m \in \mathcal{P}} y_{im} \leq 1, \quad \forall a_i \in \mathcal{A}, \quad (5)$$

$$y_{im} \in \{0, 1\}, \quad \forall a_i \in \mathcal{A}, \forall p_m \in \mathcal{P}. \quad (6)$$

Here,  $\alpha_j$  is the *requirement satisfaction index (RSI)* for task  $t_j$ , which should be equal to or greater than one to comply with the task's requirement (i.e., Equation (4)).  $y_{im}$  is a binary variable that is one if agent  $a_i$  is assigned to position  $p_m$  or zero otherwise. Equation (5) indicates that, because every agent has limited work resource, it can not execute more than two tasks in a mission and thus can be assigned to one working position at most. For fair allocation, the objective of this problem is to find an assignment set  $\{y_{im}\}$  maximising the minimum value of  $\alpha_j$ . Work capacity  $w_{im}$  is defined by Equation (1), where  $c_{im}$  is set by agent  $a_i$ 's cost-to-go function  $f_c^i(d_{im})$ , which is an increasing function with regard to the corresponding travelling distance  $d_{im}$ , i.e.,

$$c_{im} := f_c^i(d_{im}). \quad (7)$$

The agent regards  $d_{im}$  as the length of the shortest collision-free trajectory towards position  $p_m$  as follows:

$$d_{im} := \min_{\mathbf{u}_i} \int_{t_0}^{t_f} \|G\dot{\mathbf{x}}_i(t)\| dt, \quad (8)$$

subject to

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}(\mathbf{x}_i(t)) + B\mathbf{u}_i(t), \quad \forall t \in [t_0, t_f], \quad (9)$$

$$\mathbf{x}_i(t_0) = \mathbf{x}_{i,0}, \quad (10)$$

$$\mathbf{x}_i(t_f) = [p_m; \mathbf{0}_{n_d \times 1}], \quad (11)$$

$$\|G(\mathbf{x}_i(t) - \mathbf{x}_l(t))\| \geq r_{col}, \quad \forall t \in [t_0, t_f], \forall a_l \in \mathcal{A} \setminus \{a_i\}, \quad (12)$$

$$\|G\mathbf{x}_i(t) - o_q\| \geq r_{obs,q}, \quad \forall t \in [t_0, t_f], \forall o_q \in \mathcal{O}, \quad (13)$$

$$\|\mathbf{u}_i(t)\| \leq U_{max}, \quad \forall t \in [t_0, t_f], \quad (14)$$

$$\|H\mathbf{x}_i(t)\| \leq V_{max}, \quad \forall t \in [t_0, t_f], \quad (15)$$

where  $\mathbf{x}_i(t) = [\mathbf{p}_i(t); \dot{\mathbf{p}}_i(t)]$ ;  $\mathbf{p}_i(t) \in \mathbb{R}^{n_d \times 1}$  and  $\mathbf{u}_i(t) \in \mathbb{R}^{n_d \times 1}$  are the agent's position vector and control vector at time  $t$ , respectively;  $\mathbf{f}(\cdot)$  represents its motion dynamics; and  $\mathbf{x}_{i,0}$  is the given initial state at the initial time  $t_0$ . Here,  $G = [I_{n_d \times n_d} \ \mathbf{0}_{n_d \times n_d}]$ ,  $H = [\mathbf{0}_{n_d \times n_d} \ I_{n_d \times n_d}]$ ,  $B =$

$[\mathbf{0}_{n_d \times n_d} \ I_{n_d \times n_d}]^\top$ . Each agent needs to keep the collision-avoidance distance  $r_{col}$  from other agents and  $r_{obs,q}$  from obstacle  $o_q$  while moving to the assigned position (i.e., Equations (12)–(13)). Moreover, the collision-free trajectory should be feasible in terms of the agent's dynamics and physical constraints such as its available maximum control power  $U_{max}$  and velocity  $V_{max}$  (i.e., Equations (14)–(15)).  $\square$

**Assumption 1.** (The number of agents). Given an instance of Problem 1, the number of the agents  $n_a$  is large enough such that there is at least a feasible agent-position assignment that satisfies every task's minimum workload requirement, i.e.,  $\alpha_j \geq 1$  for  $\forall t_j \in \mathcal{T}$ . Otherwise, there is no solution for the instance.

**Assumption 2.** (Obstacle modelling). Each obstacle is enclosed with a circle (or sphere) with a radius of  $r_{ops}$ . Note that a complex-shaped obstacle can be addressed by introducing multiple circle-shaped obstacle avoidance areas.

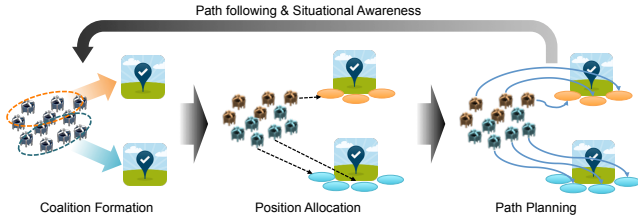
**Assumption 3.** (Robot capability). Each robot can know its position and follow the trajectory obtained from the proposed framework. The robot is also capable of hovering such as quadrotors.

### 3 Decoupling to subproblems: coalition formation, position allocation, and path planning

Our idea is to split the original problem into three subproblems, i.e., *coalition formation*, *position allocation*, and *path planning* problems, and to make the agents address the subproblems sequentially and repeatedly along with a finite time-horizon, as illustrated in Figure 2, in a decentralised manner. In the first phase, the agents partition themselves into disjoint task-specific coalitions. Once an agreed set of coalitions  $\{\mathcal{S}_j\}$  are determined, every agent in  $\mathcal{S}_j$  selects a working position amongst all the available positions for task  $t_j$ , i.e.,  $p_m \in \mathcal{P}_j$ . After that, all the agents generate and follow collision-free trajectories towards the selected working positions. The entire process is executed again before reaching the end of the time-horizon. In this section, we will show how the original problem can be decoupled into the three subproblems.

#### 3.1 Coalition formation problem

Firstly, we introduce a decision variable  $x_{ij} \in \{0, 1\}$ , which equals to one if agent  $a_i$  joins coalition  $\mathcal{S}_j$  and zero otherwise. It is implied from  $x_{ij} = 1$  that the agent will be



**Figure 2.** The proposed framework consists of three phases: coalition formation (Problem 2), position allocation (Problem 3), and path planning (Problem 4).

allocated to one of the positions for the task (i.e.,  $\forall p_m \in \mathcal{P}_j$ ). From this, it turns out that  $\sum_{\forall p_m \in \mathcal{P}_j} y_{im} = 1$ . Accordingly, it can be said that

$$x_{ij} = \sum_{\forall p_m \in \mathcal{P}_j} y_{im}. \quad (16)$$

We also introduce *abstracted cost* of agent  $a_i$  for task  $t_j$ , denoted by  $\bar{c}_{ij}$ , which abstracts all the costs towards the task's working positions, i.e.,  $\bar{c}_{ij} \approx c_{im}$  for  $\forall p_m \in \mathcal{P}_j$ . The abstracted cost is defined by  $\bar{c}_{ij} := f_c^i(\bar{d}_{ij})$ , where  $\bar{d}_{ij}$  is the shortest travelling distance of agent  $a_i$  towards task  $t_j$ 's central position (denoted by  $p_{j,0}$ ) without consideration of the inter-agent collision avoidance constraints in (12). For clear explanation, Figure 3 illustrates the notations introduced.

Since the effectiveness of working positions will not be considered in this subproblem, for the moment we let  $\eta_m = 1$ . As we did for the cost, we also abstract the agent work resource in Equation (1) as *abstracted work resource*  $\bar{w}_{ij} \approx w_{im}$  for  $\forall p_m \in \mathcal{P}_j$ . Thus, the equation can be rewritten as

$$\bar{w}_{ij} = w_{i,0} - \bar{c}_{ij}. \quad (17)$$

Under the aforementioned abstractions,  $\alpha_j$  in (3) becomes equivalent to  $\bar{\alpha}_j := \sum_{\forall a_i \in \mathcal{A}} \bar{w}_{ij} x_{ij} / R_j$ , which can be derived by substituting  $w_{im}$  with  $\bar{w}_{ij}$  and using Equation (16). Eventually, the original problem is reduced to a coalition formation problem defined as follows:

**Problem 2.** (Coalition formation for fair resource allocation).

$$\max_{\{x_{ij}\}} \left( \min_{\forall t_j \in \mathcal{T}} \bar{\alpha}_j \right), \quad (18)$$

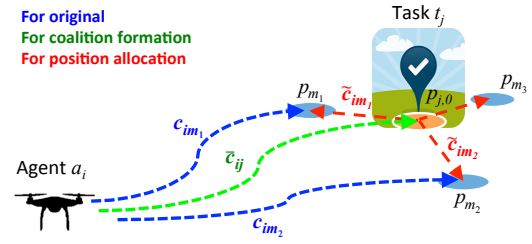
where

$$\bar{\alpha}_j := \frac{\sum_{\forall a_i \in \mathcal{A}} \bar{w}_{ij} x_{ij}}{R_j}, \quad (19)$$

subject to

$$\bar{\alpha}_j \geq 1, \quad \forall t_j \in \mathcal{T}, \quad (20)$$

$$\sum_{\forall t_j \in \mathcal{T}} x_{ij} \leq 1, \quad \forall a_i \in \mathcal{A}, \quad (21)$$



**Figure 3.** An illustration of  $c_{im}$ ,  $\bar{c}_{ij}$ , and  $\tilde{c}_{im}$ . They are used for the original problem, the coalition formation subproblem, and the position allocation subproblem, respectively.

$$x_{ij} \in \{0, 1\}, \quad \forall a_i \in \mathcal{A}, \forall t_j \in \mathcal{T}. \quad (22)$$

□

### 3.2 Position allocation problem

After forming coalitions, agents in each coalition  $\mathcal{S}_j$  have to decide working positions amongst  $\mathcal{P}_j$  to execute the assigned task  $t_j$ . In this phase, we consider the work efficiency of working positions  $\eta_m$ , which was not considered in the coalition formation problem. This agent-position allocation problem, which is the one-to-one assignment of independent single-position-required agents to independent single-agent-required positions, can be formulated by a linear assignment problem from the combinatorial optimisation literature<sup>21</sup>:

**Problem 3.** (Position allocation). Given coalition  $\mathcal{S}_j$  and a set of positions  $\mathcal{P}_j$ , the objective is to find an assignment such that

$$\max_{\{y_{im}\}} \sum_{\forall a_i \in \mathcal{S}_j} \sum_{\forall p_m \in \mathcal{P}_j} \tilde{w}_{im} y_{im} \quad \text{subject to} \quad (23)$$

$$\sum_{\forall p_m \in \mathcal{P}_j} y_{im} = 1, \quad \forall a_i \in \mathcal{S}_j, \quad (24)$$

$$\sum_{\forall a_i \in \mathcal{S}_j} y_{im} \leq 1, \quad \forall p_m \in \mathcal{P}_j, \quad (25)$$

$$y_{im} \in \{0, 1\}, \quad \forall a_i \in \mathcal{S}_j, \forall p_m \in \mathcal{P}_j. \quad (26)$$

Here,  $\tilde{w}_{im}$  is the expected available work resource of agent  $a_i$  when it arrives at position  $p_m \in \mathcal{P}_j$  via the corresponding task's central position  $p_{j,0}$ , defined by

$$\tilde{w}_{im} := \eta_m (\bar{w}_{ij} - \tilde{c}_{im}), \quad (27)$$

where  $\tilde{c}_{im} := f_c^i(\|p_m - p_{j,0}\|)$ . □

### 3.3 Path planning problem

In the last phase, given the position resulted by solving Problem 3, denoted by  $p_i^*$  for agent  $a_i$ , the agent needs

to generate a collision-free trajectory that minimises its travelling distance, as follows:

**Problem 4.** (Path planning with collision avoidance). For each agent  $a_i \in \mathcal{A}$ ,

$$\min_{\mathbf{u}_i} \int_{t_0}^{t_f} \|G\dot{\mathbf{x}}_i(t)\| dt, \quad (28)$$

subject to (9), (10), (12)–(15), and

$$\mathbf{x}_i(t_f) = [p_i^*; \mathbf{0}_{n_d \times 1}]. \quad (29)$$

□

### 3.4 Assumptions

The followings are assumptions considered in this study.

**Assumption 4.** (Agents' network). The communication network of the agents is at least strongly connected as well as satisfies the property in Assumption 10, which will be described in Section 6. Given the network at time  $t$ ,  $\mathcal{N}_i(t)$  denotes a set of neighbours nearby agent  $a_i$ , i.e.,  $\mathcal{N}_i(t) = \{a_l \in \mathcal{A} \mid \|G(\mathbf{x}_i(t) - \mathbf{x}_l(t))\| \leq R_{comm}\}$ , where  $R_{comm}$  is the agent's communication radius.

**Assumption 5.** (Accessible information). Every agent  $a_i$  can sense every neighbour agent  $a_l \in \mathcal{N}_i(t)$ , and obtain its local information such as  $\bar{c}_{lj}$ ,  $\bar{w}_{lj}$ ,  $\mathbf{x}_l$  within a reasonably short time. The agent also knows the mission scenario information regarding tasks, working positions and obstacles, e.g.,  $R_j$ ,  $\mathcal{P}$ ,  $\eta_m$ ,  $\mathcal{O}$ .

## 4 Coalition Formation

### 4.1 Algorithm

By letting  $\bar{\alpha}_{\min} := \min_{\forall t_j \in \mathcal{T}} \bar{\alpha}_j$  (called *the minimum RSI*), it can be said that the objective of Problem 2 is to find the maximised  $\bar{\alpha}_{\min}$ , denoted by  $\bar{\alpha}_{\max \min}$ . Supposing that we already know the value, the corresponding optimal assignment  $\{x_{ij}\}$  complies with the following conditions:

**Condition 1.** All the given agents join to any coalition (i.e.,  $\sum_{\forall t_j \in \mathcal{T}} x_{ij} = 1, \forall a_i \in \mathcal{A}$ ).

**Condition 2.** Given  $\bar{\alpha}_{\max \min}$ , it holds that

$$\bar{\alpha}_j \geq \bar{\alpha}_{\max \min} \quad \forall t_j \in \mathcal{T}.$$

**Condition 3.** Within constraints (21)–(22) and Condition 2, the total work capacities of the agents (i.e.,  $\sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{w}_{ij} x_{ij}$ ) are nearly maximised. Equivalently, from the definition of  $\bar{w}_{ij}$  in (17), it can be also said that the total cost of the agents (i.e.,  $\sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{c}_{ij} x_{ij}$ ) is almost minimised. Note that the optimal assignment  $\{x_{ij}\}$  for the max-min optimisation is not necessarily optimal for maximisation of the total work capacities. However, the extent of the degradation is expected to be not significant due to Condition 2 and the supposition that  $\bar{\alpha}_{\max \min}$  was already maximised.

Our fundamental approach towards Problem 2 is to search the maximised value  $\bar{\alpha}_{\max \min}$  and its corresponding assignment set that meets all the conditions. Assuming that a certain value of  $\bar{\alpha}_{\max \min} \geq 1$  is given as the nominal value, we will firstly find an assignment that satisfies Condition 2 while minimising the total cost of the agents (i.e., Conditions 3). This subproblem, called *MinGAP-MR* (*Minimisation version of General Assignment Problem with Minimum Requirements*)<sup>22</sup>, can be rewritten as Problem 5.

**Problem 5.** (Minimisation version of Generalised Assignment Problem with Minimum Requirements (MinGAP-MR)). Given a nominal value of  $\bar{\alpha}_{\max \min}$ ,

$$\min_{\{x_{ij}\}} \sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{c}_{ij} x_{ij} \quad (30)$$

subject to (21), (22), and for  $\forall t_j \in \mathcal{T}$

$$\sum_{\forall a_i \in \mathcal{A}} \bar{w}_{ij} x_{ij} \geq \bar{\alpha}_{\max \min} R_j. \quad (31)$$

□

Note that Condition 2 can be transformed to Equation (31) by the definition of  $\bar{\alpha}_j$ , i.e., Equation (19). If the resultant assignment also conforms with Condition 1, the nominal value is a suboptimal objective function value of Problem 2. If either of Conditions 1 or 2 fails, the nominal value  $\bar{\alpha}_{\max \min}$  needs to be changed, and this process is iteratively executed until both conditions are fulfilled.

Of importance is how to search  $\bar{\alpha}_{\max \min}$  properly and quickly. To this end, we adopt the concept of binary search<sup>23</sup>. Firstly, we set the nominal value of  $\bar{\alpha}_{\max \min}$  as its possible maximum value  $J_{LHR}$ , which is defined in the following proposition.

**Proposition 1.** *The optimal objective function value of Problem 2 is upper bounded by its Linear-and-Homogeneous Relaxation (LHR) objective function value, which is defined*

by

$$J_{LHR} := \frac{\sum_{\forall a_i \in \mathcal{A}} (\max_{\forall t_j \in \mathcal{T}} \bar{w}_{ij})}{\sum_{\forall t_j \in \mathcal{T}} R_j}. \quad (32)$$

**Proof.** Assume that the binary constraint (22) is relaxed so that an agent's resource can be assigned fractionally (i.e. *Linear relaxation*,  $x_{ij} \in [0, 1]$ ). The optimal solution in the relaxed problem is such that the maximised value of  $\sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{w}_{ij} x_{ij}$  is shared to the given tasks in exactly proportion to their minimum requirements. This shared value always upper bounds the optimal objective function of Problem 2. The maximised value of  $\sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{w}_{ij} x_{ij}$  is at most  $\sum_{\forall a_i \in \mathcal{A}} (\max_{\forall t_j \in \mathcal{T}} \bar{w}_{ij})$ , where each agent is assumed to have its maximum resource regardless of the tasks (i.e., (*Maximum Homogeneous relaxation*)). Combining the two relaxations gives us the value as shown in Equation (32), which also upper bounds the optimal objective function of Problem 2.  $\square$

---

**Algorithm 1** Coalition formation with fair allocation

---

- 1:  $\bar{\alpha}_{\max \min} :=$  the value of Equation (32)
  - 2:  $\beta := \bar{\alpha}_{\max \min}$
  - 3: **repeat**
  - 4:    $\beta \leftarrow \beta/2$
  - 5:    $\{x_{ij}\} :=$  the solution for Problem 5 (by Algorithm 2)
  - 6:   Modify  $\bar{\alpha}_{\max \min}$  according to Table 2
  - 7: **until** Conditions 1 and 2 are met as well as  $\beta \leq \epsilon_{CF}$
  - 8: **return**  $\{x_{ij}\}$
- 

The proposed coalition formation process is described in Algorithm 1. After setting the nominal value  $\bar{\alpha}_{\max \min} = J_{LHR}$  initially, we try to find a solution assignment for Problem 5 by Algorithm 2, which will be shown in the following section. Depending on the solution's fulfillment with regard to Conditions 1 and 2, the nominal value is modified as shown in Table 2. In the table,  $\beta$  is the amount of variation to adjust the previous nominal value.  $\beta$  is initially set by the half of  $J_{LHR}$ , and is reduced to one half at every search.

**Table 2.** How to adjust the nominal value of  $\bar{\alpha}_{\max \min}$  in Algorithm 1

Given $\{x_{ij}\}$	C2 fulfilled	C2 failed
C1 fulfilled	Increase $\bar{\alpha}_{\max \min}$	Decrease $\bar{\alpha}_{\max \min}$
C1 failed	Increase $\bar{\alpha}_{\max \min}$	Not applicable

- When increasing:  $\bar{\alpha}_{\max \min} \leftarrow \bar{\alpha}_{\max \min} + \beta$
- When decreasing:  $\bar{\alpha}_{\max \min} \leftarrow \bar{\alpha}_{\max \min} - \beta$

Failure of Condition 2 means that there exists no solution for Problem 5 with the given nominal value  $\bar{\alpha}_{\max \min}$ . Therefore, the value needs to be reduced. Failure of

Condition 1 implies that there must be another assignment that yields a higher value of  $\bar{\alpha}_{\max \min}$  exploiting all the given agents. This may be the case even if both conditions are fulfilled, and hence, we try to increase the nominal value. This iterative search is executed until the time not only when Conditions 1 and 2 are simultaneously satisfied but also when  $\beta$  is less than a certain bound  $\epsilon_{CF}$ . Note that the subroutine algorithm for Problem 5 (i.e., Algorithm 2) does not provide the case where both conditions fail, the details of which will be described in Remark 1 in the following section.

## 4.2 The subroutine for MinGAP-MR

This section addresses Problem 5 by using the distributed game-theoretical algorithm proposed in our previous study<sup>22</sup>, where the problem is modelled as a non-cooperative coalition formation game of selfish agents who have different preferences regarding task-specific coalitions. We introduce some necessary definitions and notations for the algorithm. We define a *partition* as a set  $\Pi = \{S_1, \dots, S_{n_t}, S_0\}$  that disjointly partitions the agent set  $\mathcal{A}$ . Here,  $S_0$  is the coalition of agents who do “not work any task” in  $\mathcal{T}$ . Given the partition,  $\Pi(i)$  indicates the index of the coalition to which agent  $a_i$  joins. Let  $e_{ij}$  denote the *expense* for agent  $a_i$  with regard to coalition  $S_j$  (or equivalently task  $t_j$ ). Under this algorithm, every agent tends to select the coalition requiring the lowest expense.

The objective of the algorithm is to find a *Nash stable* partition, where no agent will deviate, which is defined as follows:

**Definition 1.** (Nash-stable partition). A partition  $\Pi$  is said to be *Nash-stable* if it holds that, for every agent  $a_i \in \mathcal{A}$ ,  $e_{i\Pi(i)} \leq e_{ij}, \forall S_j \in \Pi$ .

Given  $\Pi$ , the expense  $e_{ij}$  is defined as follows:

$$e_{ij} := \begin{cases} \bar{c}_{ij+} & \text{if } a_i \in \hat{S}_j, \\ \infty & \text{otherwise,} \end{cases} \quad (33)$$

where  $\bar{c}_{ij+}$  is the *learnt cost* of agent  $a_i$  for task  $t_j$  (for now, we set  $\bar{c}_{ij+} \rightarrow \bar{c}_{ij}$ ); and  $\hat{S}_j$  is the set of agents eligible to join the coalition for task  $t_j$ . Agent  $a_i$  can ascertain its *eligibility* by an algorithm for MinKP (Minimum Knapsack Problem):  $\hat{S}_j := \text{MINKP}(S_j \cup \{a_i\}, \bar{\alpha}_{\max \min} R_j, \mathcal{W}_j, \mathcal{C}_j)^*$ , where  $S_j \in \Pi$ ,  $\mathcal{W}_j = \{\bar{w}_{kj} | \forall k : a_k \in S_j \cup \{a_i\}\}$  and  $\mathcal{C}_j = \{\bar{c}_{kj} | \forall k : a_k \in S_j\} \cup \{\bar{c}_{ij+}\}$ .

---

\*Please refer to Definition 2 in Appendix for more details. This study uses MinGreedy algorithm<sup>24</sup> as MINKP.

Since there may exist any possible divergence from a Nash-stable partition, we let each agent  $a_i$  update its learnt costs as follows:

$$\bar{c}_{ij+} \leftarrow \bar{c}_{ij+} + \lambda \cdot \bar{c}_{ij}, \quad (34)$$

where  $\lambda > 0$  is the *learning rate*, and initially  $\bar{c}_{ij+} = \bar{c}_{ij}$ . Whenever agent  $a_i$  changes its decision from coalition  $\mathcal{S}_j$  to another one, the agent learns that the previously chosen one was not suitable for itself and penalises it gradually by the learning rate as Equation (34). This can be called as *tabu learning*<sup>25</sup>. This iterative decision-making and learning process guarantees that a Nash-stable partition always can be determined<sup>22</sup>.

Its detailed procedure is described in Algorithm 2. For every agent  $a_i$ , given the current locally-known partition information  $\Pi^i$ , if the agent does not satisfy with the partition (Line 4), then it investigates every coalition  $\mathcal{S}_j \in \Pi^i$  and computes the corresponding expense  $e_{ij}$  (Lines 5–9). If there is a more preferred coalition than the existing one, the agent amends  $\Pi^i$  to reflect its new decision, updates the learnt cost for the previously-chosen one, increases  $r^i$ , which is the number of evolutions of the partition, and generates a new random time stamp  $s^i$  (Line 10–16). Then, the agent constructs a message  $M^i := \{r^i, s^i, \Pi^i, \text{satisfied}^i\}$  and sends it to other neighbour agents, and vice versa (Line 19). Using the distributed mutex algorithm<sup>26</sup> in Appendix (i.e., Algorithm 5), denoted by D-MUTEX, a single partition information can be distributedly chosen by any agent at last (Line 21). The distributed mutex algorithm enables Algorithm 2 to be executed asynchronously and distributedly as long as the network of the given agents is at least strongly-connected. Experimentally, the complexity of this algorithm is shown to be  $O(n_a n_t)$ <sup>22</sup>.

**Remark 1.** The simultaneous failure of Conditions 1 and 2 does not happen when Algorithm 2 is used. This is thanks to the definition of expense  $e_{ij}$  in Equation (33). If there exists any task  $t_j$  not satisfying the constraint (31) (i.e., Condition 2), there must be no agent who wants to join  $S_0$ . This means that all the agents join to any task-specific coalitions. Thus, Condition 1 must be fulfilled.

**Remark 2.** (How to reduce unnecessary process in Algorithm 2). We set that if agent  $a_i$  find the index of its most preferred coalition as  $j^* = 0$  (Line 8), the agent broadcasts a notification signal to other agents so that they notice Condition 1 failed. This setting enables the agents

---

**Algorithm 2** MinGAP-MR algorithm for each agent  $a_i$ 


---

```

// Note:  $\alpha_{\max \min}$  is given from Algorithm 1
// Initialisation
1: satisfiedi  $\leftarrow \mathbf{0}_{n_a \times 1}$ ;  $r^i \leftarrow 0$ ;  $s^i \leftarrow 0$ 
2:  $\Pi^i :=$  the current partition  $\Pi$ 
// Decision-making process begins
3: while satisfiedi  $\neq \mathbf{1}_{n_a \times 1}$  do
// Make a new decision if necessary
4:   if satisfiedi[ $i$ ] = 0 then
5:     Compute  $e_{ij}$  using (33)  $\forall \mathcal{S}_j \in \Pi^i$ 
6:      $j^* = \operatorname{argmin}_{\forall j} e_{ij}$ ;  $e^* = \min_{\forall j} e_{ij}$ 
7:     if  $e^* = \infty$  then // No coalition is preferred
8:        $j^* := 0$ 
9:     end if
10:    if  $j^* \neq \Pi(i)$  then
11:      Join  $\mathcal{S}_{j^*}$  and update  $\Pi^i$ 
12:       $\bar{c}_{i, \Pi^i(i)+} = \bar{c}_{i, \Pi^i(i)+} + \lambda \cdot \bar{c}_{i, \Pi^i(i)}$ 
13:       $r^i \leftarrow r^i + 1$ 
14:       $s^i \in \operatorname{unif}[0, 1]$ 
15:      satisfiedi  $\leftarrow \mathbf{0}_{n_a \times 1}$ 
16:    end if
17:    satisfiedi[ $i$ ] = 1
18:  end if
// Broadcast the local information to neighbour agents
19:  Broadcast  $M^i = \{r^i, s^i, \Pi^i, \text{satisfied}^i\}$  and
  receive  $M^l$  from its neighbours  $\forall a_l \in \mathcal{N}_i$ 
// Select the valid partition from all the received
  messages
20:  Collect all the messages  $\mathcal{M}_{rcv}^i = \{M^i, \forall M^l\}$ 
21:   $\{r^i, s^i, \Pi^i, \text{satisfied}^i\} := \text{D-MUTEX}(\mathcal{M}_{rcv}^i)$ 
22: end while
23: return  $\{x_{ij}\}$  corresponding to  $\Pi^i$ 

```

---

to early close Algorithm 2 and change the nominal value of  $\bar{\alpha}_{\max \min}$  according to Algorithm 1, and thus reduces possible unnecessary computation.

### 4.3 Analysis

We now discuss the properties of the proposed coalition formation algorithm.

**Proposition 2.** (Convergence of Algorithm 1). *Algorithm 1 terminates in a finite number of iterations.*

**Proof.** For the coalition formation algorithm to terminate, Line 7 in Algorithm 1 should be met. Since  $\beta$  is gradually reduced by one half, it is obvious that  $\beta \leq \epsilon_{CF}$  within a finite number of iterations. Now let us investigate if there will always be a moment when Conditions 1 and 2 are simultaneously fulfilled. Considering the properties of both conditions, it can be said that (1) Condition 1 is met if the nominal value of  $\bar{\alpha}_{\max \min}$  is within  $[\bar{\alpha}_{c_1}, \infty]$ , where  $\bar{\alpha}_{c_1} < \infty$ ; (2) Condition 2 is satisfied if the nominal value of  $\bar{\alpha}_{\max \min}$  is within  $[0, \bar{\alpha}_{c_2}]$ , where  $\bar{\alpha}_{c_2} > 0$ . From this, if



$\bar{\alpha}_{c_2} < \bar{\alpha}_{c_1}$ , then Algorithm 1 cannot terminate. However, as described in Remark 1, there is no such case, and hence  $[\bar{\alpha}_{c_1}, \bar{\alpha}_{c_2}]$  is the range when both conditions are fulfilled. It is clear that the modification procedure of the nominal value  $\bar{\alpha}_{\max \min}$  shown in Table 2 eventually makes the value converge to the range. The subroutine in Line 5 terminates within a finite time<sup>22</sup>, so does Algorithm 1. This completes the proof.  $\square$

**Proposition 3.** (The number of iterations in Algorithm 1). *Suppose that, after  $\kappa$  iterations of the main loop of Algorithm 1 (i.e., Lines 3–7), the nominal value  $\bar{\alpha}_{\max \min}$  remains the range where Conditions 1 and 2 are both fulfilled. Then, the maximum number of iterations happened is upper bounded by  $\max\{\kappa, \lceil \log_2(J_{LHR}/\epsilon_{CF}) \rceil\}$ .*

**Proof.** As  $\beta$  reduces to its half after each iteration, it can be said that  $\beta = J_{LHR} \cdot (\frac{1}{2})^n$ , where  $n$  is the number of iterations happened. Until the time when  $\beta \leq \epsilon_{CF}$ , the required number of iterations is  $\lceil \log_2(J_{LHR}/\epsilon_{CF}) \rceil$ . For Algorithm 1 to finish, Conditions 1 and 2 additionally need to be satisfied. Hence, the maximum number of iterations is upper bounded by  $\max\{\kappa, \lceil \log_2(J_{LHR}/\epsilon_{CF}) \rceil\}$ .  $\square$

## 5 Position Allocation

After the coalition formation process, agents in each coalition  $\mathcal{S}_j$  have to choose working positions amongst  $\forall p_m \in \mathcal{P}_j$  to execute the assigned task  $t_j$  (i.e., Problem 3). Since this problem can fall into ST-SR case in multi-robot task allocation categories<sup>8</sup>, it could be optimally solved by a linear programming (e.g., Kuhn’s Hungarian method<sup>27</sup>) in  $O(|\mathcal{S}_j||\mathcal{P}_j|^2)$  time. Note that any imbalance regarding the number of agents and positions can be addressed by including dummy agents or positions as required<sup>9</sup>. Please refer to the review papers<sup>8,9</sup> for more details.

This paper addresses Problem 3 in a more computationally efficient manner under the following assumptions:

**Assumption 6.** (Initial positions of agents from tasks). When a position allocation process begins, the distance from agent  $a_i \in \mathcal{S}_j$  to task  $t_j$ ’s central position  $p_{j,0}$  is much larger than the spatial difference from any position  $p_m \in \mathcal{P}_j$  to  $p_{j,0}$ . Hence, the costs of the agent to transition to any positions in  $\mathcal{P}_j$  are approximately the same as that to  $p_{j,0}$ , i.e.,  $c_{im} \approx \bar{c}_{ij}$ ,  $\forall p_m \in \mathcal{P}_j$ . In other words, the difference between  $p_{j,0}$  and each  $p_m \in \mathcal{P}_j$  can be negligible, and thereby it follows that

$\tilde{c}_m^j \approx 0$  in Equation (27). Hence, it can be approximated as

$$\tilde{w}_{im} := \eta_m(\bar{w}_{ij} - \tilde{c}_{im}) \approx \eta_m \bar{w}_{ij} \quad (35)$$

**Assumption 7.** (Work efficiency depending on proximity). Given any two positions  $p_{m_1}, p_{m_2} \in \mathcal{P}_j$ , if  $p_{m_1}$  is closer to task  $t_j$  than  $p_{m_2}$ , then an agent at  $p_{m_1}$  works more efficiently than one at  $p_{m_2}$ , i.e.,  $\eta_{m_1} > \eta_{m_2}$ . Likewise,  $\eta_{m_1} = \eta_{m_2}$  if  $p_{m_1}$  and  $p_{m_2}$  are identically distant from the task. This assumption can be considered reasonable because, for examples, in cooperative jamming mission<sup>7</sup> or in vision surveillance mission, an agent’s proximity brings stronger work efficiency (e.g., jamming effectiveness or sensing capability).

**Proposition 4.** *Under Assumptions 6 and 7, the optimal solution for Problem 3 is the allocation resulted by assigning higher work-capacity agents into higher work-efficiency positions.*

**Proof.** Since Problem 3 is only related to each task  $t_j$ , we firstly let  $\bar{w}_i := \bar{w}_{ij}$  for simplicity. Without loss of generality, it can be said that there are a set of agents  $\mathcal{S}_j$  such that  $\bar{w}_1 \geq \bar{w}_2 \geq \dots \geq \bar{w}_{|\mathcal{S}_j|}$ , and  $|\mathcal{S}_j|$  of working positions such that  $\eta_1 \geq \eta_2 \geq \dots \geq \eta_{|\mathcal{S}_j|}$ .

For any two numbers  $m, n \in \{1, 2, \dots, |\mathcal{S}_j|\}$  such that  $m \leq n$ ,

$$\eta_m \bar{w}_m + \eta_n \bar{w}_n \geq \eta_m \bar{w}_n + \eta_n \bar{w}_m. \quad (36)$$

This is because, considering that  $\delta := \eta_m - \eta_n \geq 0$ , the left-hand side in (36) becomes  $(\eta_n + \delta)\bar{w}_m + (\eta_m - \delta)\bar{w}_n = \eta_n \bar{w}_m + \eta_m \bar{w}_n + \delta(\bar{w}_m - \bar{w}_n)$ , and hence, this is always greater than or equal to the right-hand side in (36).

From this, it is obvious that, given any two agents in  $\mathcal{S}_j$ , it is always better to assign the higher work-resource agent to the higher work-efficiency position. According to Assumption 7, such a position is the one closer to task  $t_j$ . This also holds for multiple agents more than two. Hence, by assigning higher work-capacity agents into higher work-efficiency positions, we can find the optimal assignment for Problem 3 under Assumption 6 (i.e., Equation (35)) and Assumption 7.  $\square$

Owing to Proposition 4, a simple sorting algorithm can be utilised to find the optimal solution for the approximated Problem 3. There exist various types of sorting algorithms such as Quicksort, Mergesort, and Heapsort, and their complexities are averagely known as  $O(n \log n)$ , where  $n$  is the number of the given items to be sorted<sup>28</sup>. Since such

a sorting algorithm is too simple, in this position allocation process, every agent executes the algorithm in parallel.

## 6 Path Planning with Collision Avoidance

After the position allocation process in the previous section, the agents need to generate collision-free trajectories in a way that minimises their travelling distances, i.e., Problem 4. This paper exploits the MPC-SCP path planning algorithm<sup>20</sup>, which consists of a MPC loop and a SCP loop. Given a non-convex trajectory optimisation problem along with a nominal trajectory, the SCP loop recursively solves its approximated problem, which is convexified by the nominal trajectory as explained in the following paragraph (i.e., Problem 6), until the approximated solution is very close to the nominal one. In order to reduce the problem size, each agent considers inter-agent collision avoidance only for a time horizon from the current moment. While the myopic collision-free trajectory is being used for the time horizon, the next one is calculated before reaching the end of the horizon, which is the MPC loop. It is worth noting that the MPC-SCP algorithm can consider not only inter-agent collision avoidance but also obstacle avoidance, and that its real-time implementability was experimentally validated by using multiple quadrotors<sup>19</sup>. In this section, we briefly introduce how to utilise the MPC-SCP to Problem 4, then show the main difference of the proposed algorithm from the existing MPC-SCP.

Given the nominal trajectory of agent  $a_i$  (denoted by  $\bar{\mathbf{x}}_i$ ), we firstly linearise and discretise the problem as described in Appendix. In addition, the collision-avoidance constraints in (54) and (55) should be convex-approximated into the following affine constraints for a time horizon  $T_H$ : for every agent  $a_i \in \mathcal{A}$ ,

$$\begin{aligned} & (\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_l[k])^\top G^\top G(\mathbf{x}_i[k] - \bar{\mathbf{x}}_l[k]) \\ & \geq r_{col} \|G(\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_l[k])\|, \end{aligned} \quad (37)$$

$$k = k_0, \dots, \min\{T, k_0 + T_H\}, \forall a_l \in \mathcal{N}_i[k_0] \cap \mathcal{I}_i,$$

$$\begin{aligned} & (G\bar{\mathbf{x}}_i[k] - o_q)^\top (G\mathbf{x}_i[k] - o_q) \\ & \geq r_{obs,q} \|G\bar{\mathbf{x}}_i[k] - o_q\|, \end{aligned} \quad (38)$$

$$k = k_0, \dots, \min\{T, k_0 + T_H\}, \forall o_q \in \mathcal{O},$$

where  $\mathcal{N}_i[k]$  is the discretised-version notation of  $\mathcal{N}_i(\mathbf{t}_k)$ , and  $\mathcal{I}_i \subseteq \mathcal{A}$  is the set of agents who are more important than agent  $a_i$  meaning that agent  $a_i$  should avoid them (the formal definition of  $\mathcal{I}_i$  will be shown later in Assumption 8). The convexified constraints (37) and (38) are sufficient conditions for the original collision-avoidance constraints (54) and (55)

to hold<sup>20</sup>. Hence, Problem 4 can be reduced to the following convex-approximated problem:

**Problem 6.** (Convex program for path planning). For each agent  $a_i \in \mathcal{A}$ , given nominal trajectories  $\bar{\mathbf{x}}_l$  for  $\forall a_l \in (\mathcal{N}_i[k_0] \cap \mathcal{I}_i) \cup \{a_i\}$  at the current discretised time step  $k_0$ ,

$$\min_{\mathbf{x}_i} \sum_{k=k_0}^{T-1} \|G(\mathbf{x}_i[k+1] - \mathbf{x}_i[k])\| \quad (39)$$

subject to (51),(29),(56),(57),(37),(38).  $\square$

**Assumption 8.** (Importances of agents<sup>19</sup>). Let each agent  $a_i$  has its importance index  $\rho_i$  such that  $\rho_i \neq \rho_l$  for  $\forall l \neq i$ . For the convergence of Algorithm 3, it should be that

$$\mathcal{I}_i = \{a_l \in \mathcal{A} \mid \rho_l < \rho_i\}.$$

---

**Algorithm 3** Collision-free path planning for each agent  $a_i$

---

```

1:  $\bar{\mathbf{x}}_i[k] :=$  the solution to Problem 6 without (37),  $\forall k$ 
2: Communicate  $\bar{\mathbf{x}}_i$  to all neighbour agents  $a_l \in \mathcal{N}_i[k_0]$ 
3:  $\mathcal{K}_i := \mathcal{N}_i[k_0] \cup \{a_i\}$ 
4: while  $\mathcal{K}_i \neq \emptyset$  do
5:   if  $a_i \in \mathcal{K}_i$  then
6:      $\mathbf{x}_i[k] :=$  the solution to Problem 6,  $\forall k$ 
7:     if  $\mathbf{x}_i[k]$  is feasible then
8:        $\bar{\mathbf{x}}_{i,prev}[k] := \bar{\mathbf{x}}_i[k], \forall k$ 
9:        $\bar{\mathbf{x}}_i[k] := \mathbf{x}_i[k], \forall k$ 
10:    else
11:       $\bar{\mathbf{x}}_{i,prev}[k] := \bar{\mathbf{x}}_i[k] + \epsilon_{SCP} \cdot \mathbf{1}_{2n_d \times 1}, \forall k$ 
12:    end if
13:  end if
14:  Communicate  $\bar{\mathbf{x}}_i$  to all neighbours  $a_l \in \mathcal{N}_i[k_0]$ 
15:  if  $\|\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_{i,prev}[k]\| < \epsilon_{SCP}$  and
16:   $\|G(\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_l[k])\| > r_{col} \forall a_l \in \mathcal{N}_i[k_0] \cap (\mathcal{K}_i^c \cup \mathcal{I}_i)$ 
17:  and  $\|G\bar{\mathbf{x}}_i[k] - o_q\| > r_{obs,q} \forall o_q \in \mathcal{O}, \forall k$ , then
18:    Remove  $a_i$  from  $\mathcal{K}_i$ 
19:  end if
20:  Communicate  $\mathcal{K}_i$  to all neighbours  $a_l \in \mathcal{N}_i[k_0]$ 
21:   $\mathcal{K}_i := \mathcal{K}_i \cap (\cup_{l \in \mathcal{N}_i[k_0]} \mathcal{K}_l)$ 
22: end while
23:  $\bar{\mathbf{x}}_i$  is the approximate solution to Problem 4
```

---

The SCP loop for path planning is described in Algorithm 3. Firstly, each agent  $a_i$  sets its nominal trajectory  $\bar{\mathbf{x}}_i$  from the solution for Problem 6 without consideration of the inter-agent coalition avoidance constraint (Line 1). The agent shares the information with all the neighbours (Line 2). Let  $\mathcal{K}_i$  denote a local variable indicating neighbour agents who have not found coalition-free trajectories yet. If this is the case for agent  $a_i$ , it tries to obtain the solution for Problem 6 (Line 6). If it is feasible, then the nominal trajectory is updated. Otherwise, the previous nominal trajectory  $\bar{\mathbf{x}}_{i,prev}$

is set to any trajectory such that  $\|\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_{i,prev}[k]\| \not\leq \epsilon_{SCP}, \forall k$  (Lines 7–12). After sharing the information with all the neighbours (Line 14), agent  $a_i$  checks if the new trajectory is close enough to the previous nominal trajectory and collision-free (Lines 15–19), and updates  $\mathcal{K}_i$  (Lines 20–21).

**Remark 3.** (How to avoid infeasible solutions in Line 6). Although Problem 6 is a convex programming, there might be no feasible solution if the nominal trajectory  $\bar{\mathbf{x}}_i$  is not proper. The convexified constraints (37) and (38) based on the nominal trajectory occasionally induce the event that some agents can not generate feasible trajectories using the given maximum velocity and control powers. In this case, it is a remedy to solve Problem 6 with temporarily-reduced  $r_{col}$  or  $r_{obs,q}$  and then gradually increase them and solve the problem again. Alternatively, making the initial velocity profiles of the nominal trajectory become slowly also often works.

The main difference of Algorithm 3 from the existing work<sup>20</sup> is Lines 7–12: agents who are eventually not able to find feasible solutions for Problem 6 keep their previous trajectories, whereas other agents use new ones. When implementing the existing algorithm in the way it is (i.e., without the lines), agents who already obtained feasible trajectories have to wait to receive their neighbours' trajectories before proceeding Line 15. Although infeasible solutions can be avoided by Remark 3, this additional process might induce unnecessary waiting times for other agents in a synchronous process. Therefore, Algorithm 3 is designed such that some agents may temporarily skip their trajectory computations. We now show that, despite so, the proposed algorithm can converge to collision-free trajectories.

**Proposition 5.** (Convergence of the path planning algorithm). *Even if some agents are temporarily not able to find feasible solutions to Problem 6, Algorithm 3 can terminate within a finite time and provide conflict-free trajectories for all the agents.*

**Proof.** Lines 7–12 in Algorithm 3 imply that if agent  $a_i$  can not temporarily find a feasible solution to Problem 6, its neighbour agents maintain  $\bar{\mathbf{x}}_i$  as they knew before. For any agent amongst those neighbours, if there is a feasible solution to Problem 6, the solution is conflict-free with regard to  $\bar{\mathbf{x}}_i$ . This situation is more favourable for the agent to comply with the condition in Line 16, compared with the circumstance where all the agents find feasible solutions in

Line 6. This is because each feasible solution  $\mathbf{x}_i$  considers the corresponding neighbours' previous trajectories as the nominal values (as shown in Equation (37)), and thus the simultaneous update of every  $\mathbf{x}_i \forall a_i$  probably causes additional iterations. However, despite the fact, Morgan et al.<sup>19</sup> shows that all the agents can converge to collision-free trajectories within a finite time. Therefore, it can be said that the occurrence of temporary infeasible solutions in Line 6 does not hinder the convergence of Algorithm 3 towards conflict-free trajectories for all the agents.  $\square$

Moreover, we experimentally observed that this asynchronous process results in a faster agreement of the agents than a synchronous process, as pointed out in the work<sup>29</sup>.

## 7 The Proposed Integrated Framework

This section presents our integrated framework, as shown in Algorithm 4, that includes all the algorithm introduced previously. At the current time  $k_0$ , each agent firstly executes the coalition formation process after obtaining its abstracted costs (Lines 3–6). Note that  $\bar{d}_{ij}$  can be geometrically obtained under Assumption 2. Then, the agent proceeds the position allocation process (Lines 7–11), and the path planning algorithm (Line 12) over the time horizon  $T_H$ .

---

### Algorithm 4 Integrated decision-making framework

---

```

1:  $k_0 = 0$ ;
2: while  $k_0 \leq T - T_H$  do
  // Coalition Formation
3:   for all  $a_i \in \mathcal{A}$  (in parallel) do
4:     Compute  $\bar{d}_{ij}$  and  $\bar{c}_{ij} = f_c^i(\bar{d}_{ij})$  for  $\forall t_j \in \mathcal{T}$ 
5:   end for
6:    $\{\mathcal{S}_j\} :=$  the partition resulted by Algorithm 1
  // Position Allocation
7:   for all  $a_i \in \mathcal{S}_j, \forall t_j \in \mathcal{T}$  (in parallel) do
8:     Compute  $\tilde{w}_{im}$  using (35) for  $\forall p_m \in \mathcal{P}_j$ 
9:      $\{y_{im}\} :=$  solution to Problem 3
10:     $p_i^* = p_m$  such that  $y_{im} = 1$ 
11:   end for
  // Path planning
12:   $\mathbf{x}_i[k] :=$  the trajectories by Algorithm 3,  $\forall a_i \in \mathcal{A}$ 
  for  $\forall k \in \{k_0, \dots, k_0 + T_H\}$ 
  // Path following
13:  Follow  $\mathbf{x}_i[k]$  for  $\forall k \in \{k_0, \dots, k_0 + T_H\}, \forall a_i \in \mathcal{A}$ 
14:  Update  $k_0$  and  $\mathbf{x}_{i,k_0}$  to current time,  $\forall a_i \in \mathcal{A}$ 
15: end while

```

---

Since inter-agent collision avoidance is considered during the time horizon only, we need the following assumptions:

**Assumption 9.** (Computational feasibility). Every agent's computational and communicational capability is such that the running time of each loop of Algorithm 4 (Lines 3–12) is

less than  $T_H \Delta t$ . This guarantees that there will be no inter-agent collision during the entire mission period.

**Assumption 10.** (Detectable collisions<sup>20</sup>). To guarantee that there is no unexpected collision with other neighbour agents during the time horizon  $T_H$ ,  $R_{comm}$  and  $V_{max}$  should be such that

$$R_{comm} \geq 2V_{max}T_H\Delta t + r_{col} \text{ and } \Delta tV_{max} < r_{col}.$$

We conjecture that the integrated framework can be executed even by asynchronous behaviours of the agents to some extent. Amongst the subroutines comprising the framework, the path planning subroutine (i.e., Algorithm 3) needs a relatively more ideal environment: it requires *Local-Information Consistency Assumption (LICA)*<sup>30</sup> over neighbour agents under a communication network holding Assumptions 4 and 10. Meanwhile, our previous work<sup>22</sup> implies that the coalition formation algorithm just requires LICA under a simple strongly-connected communication network, and communication is even not necessary for the position allocation process. As long as local-information consistency required for the path planning process is guaranteed, the integrated framework can work asynchronously. Therefore, considering the fact that the path planning process is dominant, the experimental validation of the MPC-SCP<sup>19</sup> also suggests the real-time implementability of the proposed framework. More rigorous analysis regarding detrimental effects of asynchronous agents will be subject to in our future study.

For now, let us discuss about the suboptimality of the proposed integrated framework.

**Proposition 6.** (Suboptimality of the integrated framework). *Let  $r_{p_j}$  denote the maximum radius from the central position of task  $t_j$  (i.e.,  $p_{j,0}$ ) to any working positions for the task (i.e.,  $\forall p_m \in \mathcal{P}_j$ ). For an instance of Problem 2 with setting  $\bar{c}_{ij} := f_c(\bar{d}_{ij} - r_{p_j})$ , called the dummy problem, suppose that we have its LHR objective function value, denoted by  $J_{LHR}^*$ . Then, the suboptimality of an outcome from the integrated framework (i.e., Algorithm 4) is lower bounded by  $J_A/J_{LHR}^*$ , where  $J_A$  is the outcome's objective function value.*

**Proof.** Let  $J_{OPT}$  and  $J_{OPT}^*$  denote the optimal objective function value of Problem 1 and that of the dummy problem, respectively. We will firstly show that  $J_{OPT}^* \geq J_{OPT}$ . Work capacity  $\bar{w}_{ij}$  in the dummy problem is  $\bar{w}_{ij} = w_{i,0} - f_c^i(\bar{d}_{ij} - r_{p_j})$ , which is always greater than or equals to  $w_{im} = \eta_m(w_{i,0} - f_c^i(d_{im}))$  in the original problem for  $\forall p_m \in \mathcal{P}_j$ . This is because: (a) for every agent  $a_i$  and task

$t_j$ ,

$$\bar{d}_{ij} - r_{p_j} \leq d_{im}, \quad \forall p_m \in \mathcal{P}_j;$$

(b) the dummy problem has work efficiency ratio  $\eta_m = 1$ . Hence, given the optimal objective function values for both problems, it is clear that  $J_{OPT}^* \geq J_{OPT}$ .

For the dummy problem, its LHR objective function value  $J_{LHR}^*$  can be found by Equation (32). This value upper bounds  $J_{OPT}^*$  according to Proposition 1. From this, it turns out that

$$J_{LHR}^* \geq J_{OPT}^* \geq J_{OPT}.$$

Therefore, the suboptimality of the outcome from the integrated framework (i.e.,  $J_A/J_{OPT}$ ) is lower bounded by  $J_A/J_{LHR}^*$ .  $\square$

Considering the inherent urgency of a given mission, it is also possible to make the coalition formation/position allocation modules have longer cycles than the path planning module. Since this paper will apply the framework to a military mission in Section 8, we set all the modules to be executed with the same frequency.

## 8 Numerical Experiment in a Cooperative Jamming Scenario

### 8.1 Mission

We implement the proposed framework into a *cooperative stand-in jamming mission*<sup>7</sup>. This mission can be categorised as “escort jamming”, in which multiple aerial robots with ECM (electro counter measure) transmitters are to neutralise given target radars to protect their allies being far behind. The strategy for stand-in jamming differs from that for typical *stand-off* jamming in that the aerial robots penetrate into enemy territory and jam the targets while being located nearby. This comparison can be illustrated as in Figure 4. A swarm of small-sized UAVs is suitable to carry out this mission because of their small RCS (Radar Cross Section), i.e., lower observability, and fault tolerance as a multi-agent system. Despite their limited jamming resources, they can cooperatively provide enough jamming effectiveness by superimposing their signal strengths.

The cooperative jamming effectiveness by a set of  $n$  agents towards the  $j$ -th radar is a function of the *Jamming to Signal ratio* as follows<sup>31</sup>:

$$(J/S)_j = \frac{\sum_{i=1}^n J_{i,j}}{S_j}. \quad (40)$$

Here,  $S_j$  is the backscattered signal strength to the  $j$ -th radar from one of the protected allies with the largest RCS:

$$S_j = k_j^R \cdot \sigma(\theta_{a,j}) / d_{a,j}^4, \quad (41)$$

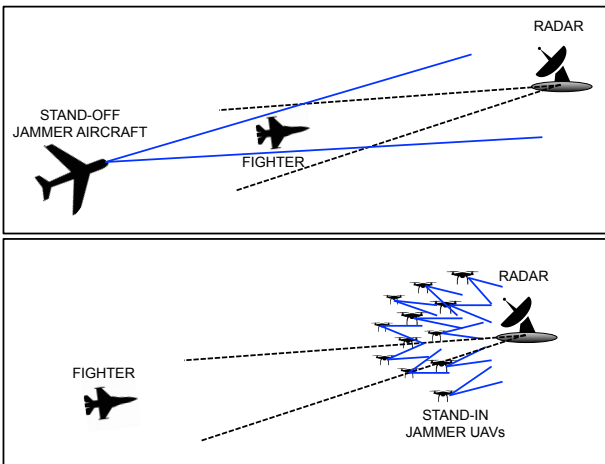
where  $k_j^R$  is the radar-dependent coefficient;  $d_{a,j}$  is the distance from the  $j$ -th radar to where the ally can be close to the radar while being protected;  $\sigma$  is the RCS of the ally, which depends on its attitude with respect to the  $j$ -th radar (denoted by  $\theta_{a,j}$ ). The individual jamming signal strength of the  $i$ -th agent towards the  $j$ -th radar is defined by

$$J_{i,j} = k_i^A \cdot G_{i,j}^R / d_{i,j}^2, \quad (42)$$

where  $k_i^A$  is the agent-dependent coefficient;  $d_{i,j}$  is the distance between the  $i$ -th agent and the  $j$ -th radar when the agent performs jamming; and  $G_{i,j}^R$  is the radar antenna gain, which relies on the agent's position with respect to the radar's main-lobe. The cooperative jamming is successful if  $(J/S)_j$  exceeds the radar's *burn-through value*  $(J/S)_j^{burn}$ , which is determined by the radar's characteristics and signal processing method.

## 8.2 Implementation and Settings

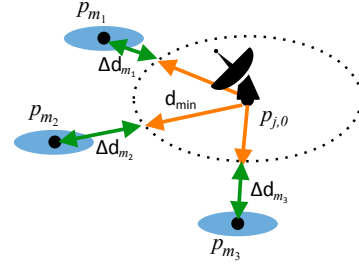
In this experiment, we have  $n_a = 50$  of UAVs,  $n_t = 3$  of tasks (i.e., target radars). For the radar and jamming-agent relevant coefficients mentioned in the previous section, the realistic values shown in Table 3 are used. Note that  $k^A$  of micro-sized UAV is assumed from consideration of other UAV types' values. We uniform randomly generate  $k_i^A \in [0.025, 0.050]$  for the agents, and set that  $\{k_1^R, k_2^R, k_3^R\} = \{2 \cdot 10^9, 1.5 \cdot 10^9, 1 \cdot 10^9\}$  for the radars.



**Figure 4.** Cooperative *Stand-off* Jamming vs. Cooperative *Stand-in* Jamming

**Table 3.** Parameters of Jamming UAVs and Radars<sup>31</sup>

UAV size	$k^A$	Radar type	$k^R$
micro	0.05	short-range	$2 \cdot 10^7$
small	0.25	med.-range	$2 \cdot 10^8$
large	1	long-range	$2 \cdot 10^9$
$(J/S)^{burn}$		$G^R$ main-lobe	$G^R$ side-lobe
1	1	1	0.001
1	1	1	0.001
1	1	1	0.001



**Figure 5.** Definitions of  $d_{min}$  and  $\Delta d_m$

We set that the agents can approach to the tasks as close to 60 m as possible, denoted by  $d_{min}$ , and they may not be located in the radars' main lobes: from (42), each agent  $a_i$ 's initially-available work resource is set by

$$w_{i,0} = k_i^A \cdot G_{side}^R / d_{min}^2, \quad (43)$$

where the corresponding radar antenna gain, denoted by  $G_{side}^R$ , is set to 0.001. Then, work efficiency ratio  $\eta_m$  for position  $p_m \in \mathcal{P}_j$  is set by

$$\eta_m = \frac{d_{min}^2}{(d_{min} + \Delta d_m)^2}, \quad (44)$$

where  $\Delta d_m$  is the shortest distance from the working position  $p_m$  to the circle surrounding the  $j$ -th radar with the radius of  $d_{min}$ , as illustrated in Figure 5. We set that as an agent uses its resource to transition its given  $k^A$  is reduced: the decreasing rate is 0.005 per km. From this, we can define each agent's cost function  $f_c^i(d)$ .

We assume that the allies to protect are F-16 fighter, whose RCS is known as  $5 m^2$ , and they should be protected until reaching 20 km before the radars. Each radar's  $(J/S)_j^{burn}$  is set to 1. Thus, each task  $t_j$ 's minimum requirement is computed by

$$R_j = S_j \cdot (J/S)_j^{burn}. \quad (45)$$

In addition, we have  $n_o = 2$  of obstacles. The inter-agent collision-avoidance radius is  $r_{col} = 15 m$ . For every obstacle, its radius is  $r_{obs,q} = 60 m$ .

For each agent  $a_i$ , we consider 2-dimensional space and the point-mass kinematics model:

$$\dot{\mathbf{x}}_i(t) = A\mathbf{x}_i(t) + B\mathbf{u}_i(t), \quad (46)$$

where

$$\begin{aligned} A &= [\mathbf{0}_{2 \times 2} \ I_{2 \times 2}; \ \mathbf{0}_{2 \times 2} \ \mathbf{0}_{2 \times 2}], \\ B &= [\mathbf{0}_{2 \times 2} \ I_{2 \times 2}]^\top, \\ \mathbf{u}_i(t) &= [u_1(t) \ u_2(t)]^\top. \end{aligned}$$

The maximum speed and acceleration of every agent is  $V_{\max} = 10 \text{ m/s}$  and  $U_{\max} = 3 \text{ m/s}^2$ , respectively.

We set that the agents generate collision-free trajectories over the time-horizon  $T_H = 30$  seconds, but update them at every 20 seconds. For the time-horizon, the discretised time gap is set to  $\Delta t = 1 \text{ sec}$ . In order to reduce computation time, we use  $\Delta t = 10 \text{ sec}$  for the time range outside  $T_H$ . It is assumed that before  $t = 0$  the agents already finished Algorithm 4 for the first time-horizon, in which they compute for the next time-horizon, and so forth. The mission final time is  $t_f = 180 \text{ sec}$ . We set that  $\epsilon_{CF} = 0.05$ ,  $\epsilon_{SCP} = 2.5$  and  $\lambda = 0.5$ .

In order to see how adaptively the agents using the proposed framework change their behaviours, we consider a dynamic environment in which some of the agents are lost in the middle of the mission: a half of agents assigned to task  $t_1$  are randomly failed at  $t = 35 \text{ sec}$ ; even so, the communication network of the remaining alive agents still holds Assumption 4; and every agent can notice the failure using the network shortly.

### 8.3 Results

Figure 6 show sequential snapshots of the resulted behaviours of the agents using the proposed framework. Each subfigure shows the locations of the agents, the tasks, the working positions and the obstacles at a certain time of the mission. The tasks are represented as blue, red, and green circles in the right, and the obstacles as black circles in the middle. The grey small circles around each task are the corresponding working positions. The size of the solid circle for each task indicates its minimum workload requirement, and that of the dashed black circle represents the minimum radius the agents can be as close to the task as possible, i.e.,  $d_{\min}$ . Each agent and its collision-avoidance radius are illustrated as a coloured dot and the circle surrounding it, respectively. Here, the circle is coloured by the colour of the agent's assigned task. The size of each dot indicates the agent's currently-available work resource, and it shrinks

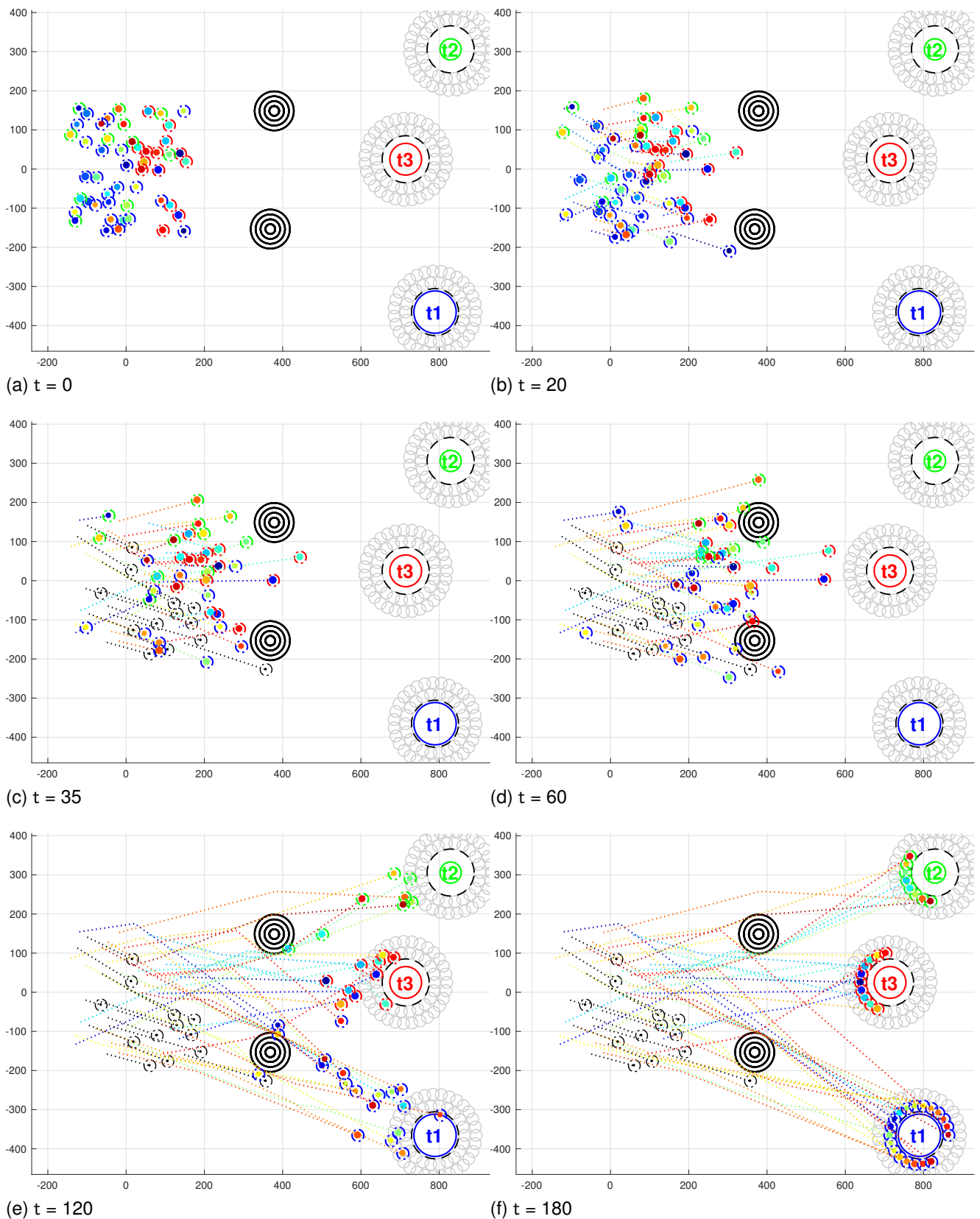
as the resource is consumed by transitioning towards its assigned working position. The dotted trail behind each agent is the agent's previous trajectory.

As presented in Figure 6(a)–(b), the agents followed the trajectories towards their assigned positions from  $t = 0$  to  $t = 20$ . At  $t = 35$ , the half of the agent assigned to task  $t_1$  were somehow failed (Figure 6(c)). Note that the lost agents and their trajectories are represented as black-dotted circles and lines, respectively. Due to the update interval for new trajectories, the remaining alive agents managed to reallocate the positions and generate new trajectories for the next time horizon by  $t = 60$  (Figure 6(c)). Here, some of agents previously assigned to tasks  $t_2$  and  $t_3$  are assigned to  $t_1$ . After that, the alive agents followed the new collision-free trajectories until the mission final time, as presented in Figure 6(e)–(f). Figure 6(f) shows that the agents were sensibly allocated with consideration of their working resources and the tasks' requirements.

Figure 7 shows each agent's minimum distance from its closest neighbour during the entire mission. Each line is coloured corresponding to the agent colour of Figure 6. This result indicates that all the agents using the proposed framework comply with the inter-agent collision avoidance constraints.

Table 4 presents the computation time spent to obtain the decision-making result. Because the framework sequentially solves an optimisation problem over the time horizon, the computation time shown in the table is the average over every optimisation. Note that the computation times spent for the coalition formation and position allocation are only significant when the agents planned over the time periods  $t \in \{[0, 20], [60, 80]\}$ , thus we only used the corresponding computation times to obtain the average value. All the simulations were performed using MATLAB R2016b on a computer (Mac mini Late 2014) with Intel Core i5 2.8 GHz, 16GB Memory and OS X Yosemite v.10.10.5. To solve convex optimisations (i.e., Problem 6) in Algorithm 3, CVX ver. 2.1<sup>32</sup> was utilised. The table shows that approximately 6 seconds are required for each time horizon. Since the agents in the experiment updated their trajectories at every 20 seconds, there still remain approximately 14 seconds extra.

In addition, Table 4 shows the suboptimality of the outcome from the proposed framework. We exploited Proposition 6 to obtain the suboptimality lower bound, which indicates that the outcome is near-optimal.

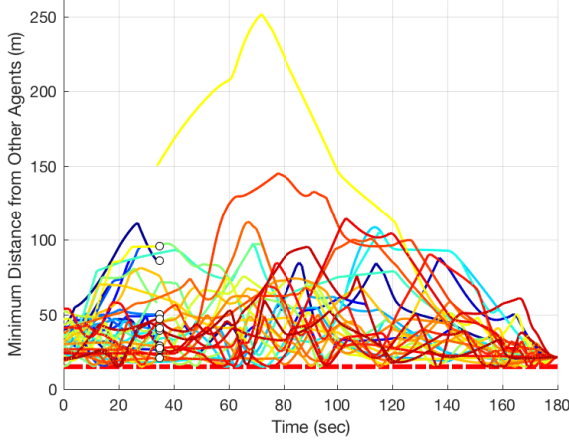


**Figure 6.** The resulted behaviours of the agents using the proposed framework ( $n_a = 50$ ,  $n_t = 3$ , and  $n_o = 2$ ).

## 9 Conclusion

This study addressed a swarm of heterogeneous robots' decision-making issues including team formation, team-to-task assignment, agent-to-working-position selection, fair

resource allocation considering tasks' minimum requirements, and trajectory optimisation with collision avoidance. The proposed framework decouples the complex original problem into three subproblems (i.e., coalition formation,



**Figure 7.** Each agent's minimum distance from its closest neighbour agent during the mission time ( $r_{col} = 15m$ ).

**Table 4.** Computation times and suboptimality of the proposed framework ( $n_a = 50$ ,  $n_t = 3$ ,  $n_o = 2$ )

Subroutine	Computation time (sec)
Coalition formation (Algorithm 1)	0.101
Position allocation	0.045
Path planning (Algorithm 3)	5.088
Suboptimality of Algorithm 4	(%)
Lower bound	97.42

position allocation, and path planning) and deals with them sequentially by three different subroutine algorithms in a decentralised manner. For the coalition formation subproblem, we introduced the game-theoretical method that recursively sets the minimum RSI and minimises given agents' unnecessary costs (equivalently maximises their work capacities). We showed that the position allocation subproblem can be solved by a simple sorting algorithm under reasonable assumptions. For the trajectory optimisation, we utilised the MPC-SCP algorithm by which the agents can generate collision-free trajectories over a time horizon. By introducing the LHR solution concept, we proposed a methodology to analyse suboptimality of the proposed framework. As a proof of concept, we implemented the proposed integrated framework into a UAV swarm's cooperative stand-in jamming mission scenario. It was suggested from the numerical experiment results that the framework could be computationally feasible, fault tolerance, and provide a near-optimal outcome.

A natural progression of this study is to validate this framework in a real-robot experiment. Furthermore, a formal analysis regarding the algorithmic complexity must be a significant contribution.

## 10 Appendix

### 10.1 Linearisation and Discretisation

In order to address Problem 4, we firstly linearise the dynamics in (9) about the nominal trajectory  $\bar{\mathbf{x}}_i$ , which is assumed to be given.

$$\dot{\mathbf{x}}_i = A(\bar{\mathbf{x}}_i)\mathbf{x}_i + B\mathbf{u}_i + z(\bar{\mathbf{x}}_i) \quad (47)$$

where  $A(\bar{\mathbf{x}}_i) = \frac{\delta \mathbf{f}}{\delta \mathbf{x}_i} |_{\bar{\mathbf{x}}_i}$  and  $z(\bar{\mathbf{x}}_i) = \mathbf{f}(\bar{\mathbf{x}}_i) - \frac{\delta \mathbf{f}}{\delta \mathbf{x}_i} |_{\bar{\mathbf{x}}_i} \bar{\mathbf{x}}_i$ . Then, we transform the problem to the discrete-time version. Regarding Equation (28), it follows from  $\|G\dot{\mathbf{x}}_i(t)\| dt = \|d\mathbf{p}_i(t)/dt\| dt$  that

$$\begin{aligned} \int_{t_0}^{t_f} \|G\dot{\mathbf{x}}_i(t)\| dt &= \int_{t_0}^{t_f} \|d\mathbf{p}_i(t)/dt\| dt \\ &\approx \int_{t_0}^{t_f - \Delta t} \left\| \frac{\mathbf{p}_i(t + \Delta t) - \mathbf{p}_i(t)}{\Delta t} \right\| dt, \end{aligned} \quad (48)$$

where  $\Delta t$  is the time difference for the discretisation. We set that, for  $k = 0, 1, \dots, T$ ,

$$\mathbf{x}_i[k] := \mathbf{x}_i(t_k), \quad \mathbf{p}_i[k] := \mathbf{p}_i(t_k), \quad \mathbf{u}_i[k] := \mathbf{u}_i(t_k), \quad (49)$$

where  $T := (t_f - t_0)/\Delta t$  is the number of discrete time steps;  $t_T := t_f$ ;  $t_{k_0} := t_0$ ; and  $t_{k+1} := t_k + \Delta t$  for all  $k$ . Then, Equation (48) becomes  $\sum_{k=k_0}^{T-1} \|\mathbf{p}_j[k+1] - \mathbf{p}_j[k]\|$ . Finally, the right term in Equation (28) becomes

$$\min_{\mathbf{u}_i} \sum_{k=k_0}^{T-1} \|G(\mathbf{x}_i[k+1] - \mathbf{x}_i[k])\|. \quad (50)$$

Likewise, Equation (47) can be reduced to

$$\begin{aligned} \mathbf{x}_i[k+1] &= A_i[k]\mathbf{x}_i[k] + B_i[k]\mathbf{u}_i[k] + z_i[k], \\ k &= k_0, \dots, T-1 \end{aligned} \quad (51)$$

where

$$\begin{aligned} A_i[k] &= e^{A(\bar{\mathbf{x}}_i(t_k))\Delta t}, \quad B_i[k] = \int_0^{\Delta t} e^{A(\bar{\mathbf{x}}_i(t_k))\tau} B d\tau, \\ z_i[k] &= \int_0^{\Delta t} e^{A(\bar{\mathbf{x}}_i(t_k))\tau} z(\bar{\mathbf{x}}_i(t_k)) d\tau. \end{aligned}$$

Equations (10), (29), (12)–(15) can be also written in discretised form as follows:

$$\mathbf{x}_i[k_0] = \mathbf{x}_{i,0} \quad (52)$$

$$\mathbf{x}_i[T] = [p^*; \mathbf{0}_{n_d \times 1}], \quad (53)$$

$$\|G(\mathbf{x}_i[k] - \mathbf{x}_l[k])\| \geq r_{col} \quad k = k_0, \dots, T, \forall a_l \in \mathcal{A}, a_l \neq a_i \quad (54)$$



$$\|G\mathbf{x}_i[k] - \mathbf{o}_q\| \geq r_{obs,q} \quad k = k_0, \dots, T, \forall o_q \in \mathcal{O} \quad (55)$$

$$\|\mathbf{u}_i[k]\|_2 \leq U_{max} \quad k = k_0, \dots, T \quad (56)$$

$$\|H\mathbf{x}_i[k]\|_2 \leq V_{max} \quad k = k_0, \dots, T \quad (57)$$

## 10.2 Minimisation Knapsack Problem (MinKP)

Our proposed approach use an algorithm for the 0/1 minimisation knapsack problem (MinKP, for short)<sup>24</sup> as its subroutine. MinKP is defined as:

**Definition 2.** *the 0/1 minimisation knapsack problem.* Suppose that there are a knapsack with its minimum requirement  $R$  and a set of  $n$  items  $Z = \{z_1, \dots, z_n\}$ , where each item  $z_i$  has its value  $v_i$  and cost  $c_i$ . The objective is to pack the knapsack with the items so that the total value of all inserted items exceeds the minimum requirement while minimising the resultant total cost:

$$\min_{\{x_i \in \{0,1\}\}} \sum_{i=1}^n c_i x_i \quad \text{s.t.} \quad \sum_{i=1}^n v_i x_i \geq R$$

where  $x_i = 1$  if item  $z_i$  is inserted in the knapsack. Let  $\text{MinKP}(Z, R, \mathcal{V}, \mathcal{C})$  denote an algorithm for MinKP, where  $\mathcal{V} = \{v_i\}$  and  $\mathcal{C} = \{c_i\}$  are sets of the items' values and costs, respectively. The output of this algorithm is the set of selected item for the knapsack.

## 10.3 Distributed Mutex Subroutine

### Algorithm 5 Distributed Mutex Subroutine<sup>26</sup>

```

1: function D-MUTEX( $\mathcal{M}_{rcv}^i$ )
2:   for each message  $M_k \in \mathcal{M}_{rcv}^i$  do
3:     if ( $r^i < r^k$ ) or ( $r^i = r^k$  &  $s^i < s^k$ ) then
4:        $r^i \leftarrow r^k$ 
5:        $s^i \leftarrow s^k$ 
6:        $\Pi^i \leftarrow \Pi^k$ 
7:        $\text{satisfied}^i \leftarrow \text{satisfied}^k$ 
8:     end if
9:   end for
10:  return  $\{r^i, s^i, \Pi^i, \text{satisfied}^i\}$ 
11: end function

```

For Algorithm 2, we use the distributed mutex algorithm proposed in our previous work<sup>26</sup>, the detail of which is described in Algorithm 5. The algorithm makes sure that there is only one (local) partition that dominates (or will finally dominate depending on the communication network) any other partitions. In other words, multiple partitions locally evolve and some of them only eventually can survive at every main loop of Algorithm 2 even under asynchronous behaviours of agents as long as their communication network is at least strongly-connected. Even if we may encounter

multiple Nash stable partitions at last, one of them can be distributedly selected by the agents.

## Funding

This work was supported by International Joint Research Programme with Chungnam National University (No. EFA3004Z).

## Acknowledgements

Thanks to Yoonsoo Kim for constructive comments.

## References

1. Sahin E. Swarm Robotics: From Sources of Inspiration to Domains of Application. In *Swarm Robotics*. Berlin: Springer, 2005. pp. 10–20.
2. Navarro I and Matía F. An Introduction to Swarm Robotics. *ISRN Robotics* 2013; 2013: 1–10.
3. Brambilla M, Ferrante E, Birattari M and Dorigo M. Swarm Robotics: a Review from the Swarm Engineering Perspective. *Swarm Intelligence* 2013; 7(1): 1–41.
4. Barton K and Kingston D. Systematic Surveillance for UAVs: A Feedforward Iterative Learning Control Approach. In *American Control Conference*. Washington, DC, USA, 2013, pp. 5917–5922.
5. Bekmezci I, Sahingoz OK and Temel S. Flying Ad-Hoc Networks (FANETs): A Survey. *Ad Hoc Networks* 2013; 11(3): 1254–1270.
6. Erdelj M, Natalizio E, Chowdhury KR and Akyildiz IF. Help from the Sky: Leveraging UAVs for Disaster Management. *IEEE Pervasive Computing* 2017; 16(1): 24–32.
7. Jang I, Jeong J, Shin HS, Kim S, Tsourdos A and Suk J. Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming. *IFAC-PapersOnLine* 2017; 50(1): 8011–8018.
8. Gerkey BP and Mataric MJ. A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems. *International Journal of Robotics Research* 2004; 23(9): 939–954.
9. Korsah GA, Stentz A and Dias MB. A Comprehensive Taxonomy for Multi-robot Task Allocation. *The International Journal of Robotics Research* 2013; 32(12): 1495–1512.
10. Shin HS and Segui-Gasco P. UAV Swarms: Decision-Making Paradigms. In *Encyclopedia of Aerospace Engineering*. John Wiley & Sons, 2014. pp. 1–13.
11. Hoy M, Matveev AS and Savkin AV. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica* 2015; 33(03): 463–497.
12. Saicharan B, Tiwari R and Roberts N. Multi Objective optimization based Path Planning in robotics using nature

- inspired algorithms: A survey. In *IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*. 2016, pp. 1–6.
13. Rubenstein M, Cornejo A and Nagpal R. Programmable Self-assembly in a Thousand-robot Swarm. *Science* 2014; 345(6198): 795–799.
  14. Shehory O and Kraus S. Methods for Task Allocation via Agent Coalition Formation. *Artificial Intelligence* 1998; 101(1-2): 165–200.
  15. Alonso-Mora J, Breitenmoser A, Ruffi M, Siegwart R and Beardsley P. Image and animation display with multiple mobile robots. *International Journal of Robotics Research* 2012; 31(6): 753–773.
  16. Turpin M, Michael N and Kumar V. CAPT: Concurrent assignment and planning of trajectories for multiple robots. *International Journal of Robotics Research* 2014; 33(1): 98–112.
  17. Turpin M, Mohta K, Michael N and Kumar V. Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots* 2014; 37(4): 401–415.
  18. Morgan D, Subramanian GP, Bandyopadhyay S, Chung SJ and Hadaegh FY. Probabilistic Guidance of Distributed Systems using Sequential Convex Programming. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Chicago, IL, USA, 2014, pp. 3850–3857.
  19. Morgan D, Subramanian GP, Chung SJ and Hadaegh FY. Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research* 2016; 35(10): 1261–1285.
  20. Morgan D, Chung SJ and Hadaegh FY. Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming. *Journal of Guidance, Control, and Dynamics* 2014; 37(6): 1725–1740.
  21. DF Votaw J and Orden A. The Personnel Assignment Problem. In *Symposium on Linear Inequalities and Programming*. Washington, DC: Planning Research Division, Comptroller, Headquarters US Air Force, 1952, pp. 155–163.
  22. Jang I, Shin HS and Tsourdos A. A Game-theoretical Approach to Heterogeneous Multi-robot Task Assignment Problem with Minimum Workload Requirements. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. 2017, pp. 156–161.
  23. Cormen TH, Leiserson CE, Rivest RL and Stein C. *Introduction to Algorithm*. 3rd ed. Cambridge, Massachusetts, USA: The MIT Press, 2009.
  24. Güntzer MM and Jungnickel D. Approximate Minimization Algorithms for the 0/1 Knapsack and Subset-Sum Problem. *Operations Research Letters* 2000; 26(2): 55–66.
  25. Beyer D and Ogier R. Tabu Learning: A Neural Network Search Method for Solving Nonconvex Optimization Problems. In *Proceedings of 1999 IEEE International Joint Conference on Neural Networks*. 1991, pp. 953–961.
  26. Jang I, Shin HS and Tsourdos A. Anonymous Hedonic Game for Task Allocation in a Large-scale Multiple Agent System. *arXiv:171106871 [csMA]* 2017; .
  27. Khun HW. The Hungarian Method for the Assignment Problem. *Naval Reserach Logistic Quarterly* 1955; 2: 83–97.
  28. Mehlhorn K and Sanders P. *Algorithms and Data Structures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.
  29. Qin Y, Cao M and Anderson BDO. Asynchronous Agreement through Distributed Coordination Algorithms Associated with Periodic Matrices. In *Proceedings of the IFAC World Congress*. Toulouse, France, 2017, pp. 1778–1783.
  30. Johnson LB, Choi HL and How JP. The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial. *IEEE Control Systems* 2016; 36(4): 45–58.
  31. Kim J and Hespanha P. Cooperative radar jamming for groups of unmanned air vehicles. In *IEEE Conference on Decision and Control*. Atlantis, Bahamas, 2004, pp. 632–637.
  32. Grant M and Boyd S. CVX: Matlab software for disciplined convex programming, ver 2.1, 2017.