J Intell Robot Syst (2011) 62:125–158
DOI 10.1007/s10846-010-9439-2

UNMANNED SYSTEMS PAPER

Multiple UAV Coalitions for a Search and Prosecute Mission

Joel G. Manathara · P. B. Sujit · Randal W. Beard

Received: 26 December 2009 / Accepted: 25 May 2010 / Published online: 16 June 2010
© Springer Science+Business Media B.V. 2010

Abstract Unmanned aerial vehicles (UAVs) have the potential to carry resources in support of search and prosecute operations. Often to completely prosecute a target, UAVs may have to simultaneously attack the target with various resources with different capacities. However, the UAVs are capable of carrying only limited resources in small quantities, hence, a group of UAVs (*coalition*) needs to be assigned that satisfies the target resource requirement. The assigned coalition must be such that it minimizes the target prosecution delay and the size of the coalition. The problem of forming coalitions is computationally intensive due to the combinatorial nature of the problem, but for real-time applications computationally cheap solutions are required. In this paper, we propose decentralized sub-optimal (polynomial time) and decentralized optimal coalition formation algorithms that generate coalitions for a single target with low computational complexity. We compare the performance of the proposed algorithms to that of a global optimal solution for which we need to solve a centralized combinatorial optimization problem. This problem is

J. G. Manathara
Department of Aerospace Engineering, Indian Institute of Science, Bangalore, 560 012, India
e-mail: joel@aero.iisc.ernet.in

P. B. Sujit (✉)
Department of Electrical and Computer Engineering, University of Porto, Porto, Portugal
e-mail: sujit@fe.up.pt

R. W. Beard
Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT 84604, USA
e-mail: beard@byu.edu

computationally intensive because the solution has to (a) provide a coalition for each target, (b) design a sequence in which targets need to be prosecuted, and (c) take into account reduction of UAV resources with usage. To solve this problem we use the *Particle Swarm Optimization* (PSO) technique. Through simulations, we study the performance of the proposed algorithms in terms of mission performance, complexity of the algorithms and the time taken to form the coalition. The simulation results show that the solution provided by the proposed algorithms is close to the global optimal solution and requires far less computational resources.

Keywords Multi UAV · Coalition formation · Task allocation · Particle swarm optimization

1 Introduction

In recent time, there has been an increase in the use of unmanned aerial vehicles (UAVs) for search-and-prosecute missions. In a typical search and prosecute mission, multiple UAVs cooperate with each other as a team to detect and prosecute targets. Deploying multiple UAVs provides robustness to the mission and reduces the mission completion time.

Often, the UAVs may have to use various types and quantities of resources to completely prosecute a target. A single UAV may not have sufficient resources, hence a sub-team of UAVs whose total resources are sufficient to completely prosecute the target needs to be assigned. This sub-team of UAVs is called a *coalition* and each agent in the coalition is called a *coalition member*. The agent that detected the target is called the *coalition leader*. Other objectives for the mission are (i) prosecute the target in minimum time, (ii) minimize the coalition size, and (iii) simultaneously prosecute the target. The objective (i) enables the UAVs to quickly accomplish the mission. The objective (ii) is required because the UAVs have limited sensor range, therefore they need to distribute the search effort to detect the target. A small coalition will allow higher numbers of agents for search task, while a larger coalition will allow fewer agents for each search task, thus reducing the total search effort and increasing the mission time. The objective (ii) implicitly assists in achieving the mission quickly. The objective (iii) is required to induce maximum damage to the target. Additionally, the resources of the UAVs deplete with use and the UAVs have kinematic constraints. The coalition formation algorithms are computationally intensive and are NP-Hard. Since, the UAVs are in motion, the algorithms need to have low computational overhead. Therefore, there is a need to develop coalition formation algorithms that have low computational complexity, satisfy the objectives (i)–(iii), and take the depletion of resources and kinematic constraints into account.

Grekey and Mataric [1], present a taxonomy for the multi-robot task allocation problem. Our problem falls in the Multi Task-Multi Robot (MT-MR) instantaneous assignment problem category according to their taxonomy because the UAVs can perform search and target prosecution tasks simultaneously and the targets may require multiple UAVs to prosecute. The problem of allocating tasks to UAVs is similar to the multi-robot task allocation problem. However, there are two differences. First, the UAVs travel with greater velocities than ground robots. Second, the resources of each robot do not deplete with use. Thus, the multi-robot task allocation

algorithms cannot be directly used. Many researchers have developed task allocation algorithms for efficiently allocating UAVs to different tasks. Most of the solutions to task allocation problems assume the UAVs (a) are homogenous, (b) can carry resources that do not deplete with use, (c) can individually prosecute the target, and (d) can prosecute the target with any resource [2–7]. Therefore, we cannot apply these algorithms directly to the problem that we consider in this paper.

A coalition is a group of team members that have agreed to cooperate with each other to execute a single task [8]. The coalitions formed are temporary by nature; once the task is accomplished, the coalition members can perform other tasks. Determining the optimal coalition from a group of agents is a computationally intensive task and is NP-hard [9]. However, there are algorithms that provide approximate and near-optimal solutions [11].

Forming a coalition to achieve tasks is an active field of research in the multi-agent community [9, 10] and the multi-robot community [12–14]. The coalition formation algorithms developed in the multi-agent community cannot be directly applied to multiple robot systems since the resources cannot be transferred from one robot to another [11]. Vig and Adams [12, 13] develop a coalition formation scheme, where the tasks act as agents and perform the function of an auctioneer for gathering bids and determining the coalition. This process of forming coalitions is different from the typical approach where a robot is an auctioneer. In [12], the authors use a negotiation process to form the coalition. It is well known that the negotiation process requires significant amount of communication and also takes time to form the coalition. The authors of [13] pose the problem as a matching problem that is also computationally intensive. The same authors address the issue of balance in the coalition between the coalitional size and the resource contribution in [11]. The algorithms developed in [12, 13], or [11] require significant amount of computation time and communication which may not be possible in UAV networks as UAVs move fast and cannot stop in mid air. Hence, these algorithms cannot be directly applied to the UAV scenario. Parker and Tang [14] present a coalition formation scheme where a coalition leader robot broadcasts the existence of a task and other robots reply by providing their availability. The leader robot evaluates all possible coalitions and sends an accept decision to the robots that it considers suitable. The task is executed by sharing the sensor information. However, they do not take coalition size or minimum arrival time to the target into account.

The problem of forming a coalition has also not been adequately addressed in the UAV community. Kingston and Schumacher [22] assign multiple UAVs to track and prosecute a target using an mixed integer linear programming (MILP) formulation. The goal is to minimize the length of task tours of the UAVs with known target positions. The resources of the tracking agents have the same capability and the problem of depleting resources is not addressed. In our problem, we account for depleting resources and assume that the agents have non-uniform payload of resources.

In our problem, we assume the UAVs have limited sensor range and search for targets in a confined region. Once an agent detects a target, it broadcasts the information about the kind of resources required to prosecute the target. In this paper,we assume that the communication is global. The UAVs that have at least one of the required resources send their information to the detecting agent. The detecting agent determines a coalition of agents to prosecute the target. To determine the coalition, we develop polynomial time sub-optimal and optimal coalition formation

algorithms. Once the coalition is formed, the agents coordinate with each other by modifying their paths such that they attack the target simultaneously.

The multiple agent rendezvous problem addresses the issue of multiple agents arriving simultaneously to a common location by cooperating with each other [15–17]. The stability of these solutions can be achieved using concepts from graph theory. McLain and Beard [18] determine a time when the rendezvous should take place and using a consensus algorithm they achieve the rendezvous for multiple UAVs. Furukawa et al. [19], considered a search and destroy mission where the target location, coalition leader and the number of UAVs required to form a coalition are predefined. The coalition leader searches for other UAVs to form a coalition and simultaneously attack the target by generating commands given by a time-optimal control problem. Notarstefano and Bullo [20] present a distributed rendezvous algorithm for multiple agents connected in a network using consensus algorithms. The simultaneous arrival scheme that we develop in this paper is different than these approaches. In our approach, the coalition leader determines the agent with the latest arrival time at the target and establishes this time as the arrival time of the coalition. Accordingly, the coalition members replan their paths to meet this constraint. In our approach, the agents do not communicate with each other and have only one communication with the leader, thus reducing communication costs.

We need to establish a benchmark for comparing mission performance of our algorithms. In order to determine the deviation of the mission from the global optimal mission where the target locations with their resource requirements, and the UAV initial positions with their resources are known *a priori*, we solve the global optimization problem. Finding the globally optimal solution is computationally intensive and difficult as we need to determine both the coalition for each target and the sequence in which the targets need to be prosecuted, while accounting for depletion of the resources. *Linear Programming* based techniques may not deliver the solution taking all the constraints of the problem. Hence we use the *Particle Swarm Optimization* technique (PSO) that can produce a global optimal solution and meet the constraints.

The rest of the paper is organized as follows. The mission scenario and the problem formulation is described in Section 2. The proposed coalition formation algorithm is presented in Section 3. For comparison purposes, we solve the combinatorial optimization problem using particle swarm optimization which is presented in Section 4. Monte-Carlo simulation results are presented in Section 5, and the conclusion is presented in Section 6.

2 Problem Formulation

A search and prosecute mission is carried out on a battlefield scenario using N heterogeneous UAVs (also called agents). The UAVs are heterogeneous as they can carry different types of resources in limited quantities. Some of these resources are consumable, while others like sensors do not deplete with use. Each UAV has a unique token number $A_i, i = 1, \ldots, N$ that is assigned prior to the start of mission. We assume that there are M targets whose initial positions are unknown. The UAVs have the capacity to carry different types of resources. Assume that

UAV A_i can carry n types of resources represented by a capability vector $\mathcal{R}^{A_i}$ of the form:

$$\mathcal{R}^{A_i} = \left(R_1^{A_i}, \ldots, R_n^{A_i}\right), \quad i = 1, \ldots, N \tag{1}$$

where $R_p^{A_i}$, $p = 1, \ldots, n$ represents the number of type-p resources held by agent A_i. For example, $\mathcal{R}^{A_i} = (4, 2, 0, 6)$ implies that agent A_i has four type-1 resources $\left(R_1^{A_i} = 4\right)$, two type-2 resources $\left(R_2^{A_i} = 2\right)$, zero type-3 resources $\left(R_3^{A_i} = 0\right)$, and six type-4 resources $\left(R_4^{A_i} = 6\right)$.

The UAVs have limited sensor range and do not have *a priori* knowledge of the target locations and their resources. To detect the targets, the UAVs have to carry out a search task. When A_i detects a target T_j, we assume that the agent can also determine the type of resources required to prosecute the target. If m-different types of resources are required to engage target T_j, then the resource requirement vector is represented as

$$\mathcal{R}^{T_j} = \left(R_1^{T_j}, \ldots, R_m^{T_j}\right), \quad j = 1, \ldots, M \tag{2}$$

where $R_q^{T_j}$, $q = 1, \ldots, m$ represents the quantity of type-q resources required to prosecute the target T_j. We assume that $m = n$, that is, the agents can carry all types of resources that are required by the target. For example: $\mathcal{R}^{T_j} = (3, 0, 5)$ indicates that to prosecute target T_j, the agents need three type-1 resources $\left(R_1^{T_j} = 3\right)$, zero type-2 resources $\left(R_2^{T_j} = 0\right)$, and five type-3 resources $\left(R_3^{T_j} = 5\right)$.

We assume that each agent can communicate with other agents in the operating zone. Once a target is located and identified, agent A_i broadcasts the target resource requirement vector $\mathcal{R}^{T_j}$ to the other agents. The UAVs in the search region possessing at least one of the required resources to strike the target T_j will respond to A_i with their cost and resource capabilities. The cost is based on the earliest time of arrival (ETA) of the agent at the target location using Dubins curves. The Dubins curves determines the shortest path between a given start and goal locations taking the kinematic constraints of the vehicle into account [21].

Assume that the UAV team takes $\mathcal{T}$ time units to accomplish a mission by prosecuting all the targets. The mission time $\mathcal{T}$ depends on the time taken by the agents to detect the targets which in turn depends on the performance of the search strategy and the time taken by the coalitions to prosecute the assigned target. The objective is to accomplish the mission in minimum time. That is

$$\text{Global Objective} : \min \mathcal{T} \tag{3}$$

Here $\mathcal{T} = f\left(\mathcal{T}_s, \sum_{j=1}^{M} \mathcal{T}_c^{T_j}\right)$, where $\mathcal{T}_s$ is the search time and $\mathcal{T}_c^{T_j}$ represents the time taken by the coalition to prosecute the target T_j. It is to be noted that the search time $\mathcal{T}_s$ implicitly depends on the coalition formation strategies. This is because, for a given target and for a given search strategy, out of two coalitions with same prosecution time, the coalition with lower size will make more UAVs available for search which may lead to a reduction in the total search time. In this paper, we

attempt to achieve the global objective (Eq. 3) by minimizing the time taken by the coalitions to prosecute all the targets, that is,

$$\min \sum_{j=1}^{M} \mathcal{T}_c^{T_j} \tag{4}$$

and by minimizing the size of coalitions formed to prosecute each target. Also, a simultaneous attack is desirable to successfully prosecute the target. Thus, the task of a coalition leader is to form a coalition such that (i) the target is prosecuted in minimum time, (ii) the coalition contains minimum number of UAVs, and (iii) the coalition members simultaneously prosecute the target. The objective (i) (Minimizing the time-to-attack the target) is a desirable quantity in achieving the objective given in Eq. 4, the objective (ii) (which is to minimize coalition size) ensures that the UAVs distribute their search effort so that the targets are detected quickly and the mission is accomplished faster, and the objective (iii) maximizes the element of surprise.

The role of the coalition leader A_i is to form a coalition C_j^i (with combined resources $\mathcal{R}_j^{C^i} = \sum_{A_k \in C_j^i} \mathcal{R}^{A_k}$) for the target T_j taking the constraints (i)–(iii) into account. Assume a worst case scenario where all the N agents are assigned to be part of a coalition. For this case, let $\mathcal{D}_i^j = \{D_1^j, \cdots, D_N^j\}$ represent the set of earliest time of arrival for the N agents, and D_k^j denotes the earliest time of arrival (ETA) of the agent A_k to arrive at the location of target T_j using Dubins curves [21]. The N agents are represented in a set $\mathcal{I}_A$, $D_k^j \in \mathcal{D}_i^j$ is the ETA of agent $A_k \in \mathcal{A}_j$ and we define Π_A as the set of all possible index sets over N agents. The ETA is used as the cost metric. To determine a coalition, the coalition leader solves the objective function:

$$\min_{\mathcal{I}_A \in \Pi_A} \max_{A \in \mathcal{I}_A} D_A^j \tag{5}$$

$$\text{subject to} \quad \left(\sum_{A \in \mathcal{I}_A} \mathcal{R}_p^A\right) \geq R_p^{T_j}, \quad \text{for all } p = 1, \ldots, m \tag{6}$$

where $\max_{A \in \mathcal{I}_A}$ represents $\mathcal{T}_c^{T_j}$ the latest arrival time of agents in $\mathcal{I}_A$. The $\min_{\mathcal{I}_A \in \Pi_A}$ part of the objective function minimizes the coalition size satisfying Eq. 6. This set satisfies the objectives (i) and (ii). Minimizing the $\mathcal{T}_c^{T_j}$ for every target T_j will minimize the objective function given in Eq. 4. Also, the coalition members have to prosecute the target simultaneously (objective (iii)). In order to prosecute simultaneously, the agents have to arrive at the target location at the same time. The minimum time (objective (i)) for the coalition to arrive at the target is determined by the agent that has the highest ETA (latest arrival time), that is, $\mathcal{T}_c^{T_j}$. As we need to determine both the smallest coalition and the coalition that can prosecute that target in minimum time, the objective function is coupled. This coupling makes the optimization problem nontrivial to be solved using standard techniques.

The solution to the optimization problem (Eqs. 5 and 6) can be found by searching the complete solution space which is computationally intensive. The computational effort exponentially increases with the number of agents and targets. In this paper, we reduce the complexity of the solution by decomposing the problem into two-stages. In the first stage, we will determine a minimum time coalition set, and using this set we determine the smallest coalition that satisfies the constraints. We develop both a suboptimal polynomial time algorithm as well as an optimal algorithm. In the next section, we describe the two-stage coalition formation algorithms.

The agent that detects a target becomes the coalition leader and forms the coalition. Hence, the coalition formation algorithms are decentralized by nature which has significant advantages over centralized solution concepts. However, if the target locations and their resources, with the UAV positions and their resources are known *a priori*, then we can solve the global optimization problem off-line. The global optimization problem has to determine the appropriate coalitions for each target taking the reduction of the UAV resources into account. The global solution should also determine the order in which targets are prosecuted. In order to find a solution to this highly complex problem and to take the dynamics of the resources into account, we propose a particle swarm optimization (PSO) based solution which we describe in Section 4.

The UAVs are subjected to kinematic constraints preventing instantaneous course changes. We assume that the autopilots of the UAVs hold a constant altitude and ground speed. The kinematics of the i^{th} UAV is therefore modeled using first order kinematics as

$$\begin{aligned} \dot{x}_i &= v_i \cos\psi_i, \\ \dot{y}_i &= v_i \sin\psi_i, \\ \dot{\psi}_i &= k(\psi_i^d - \psi_i), \qquad i = 1, \ldots, N \end{aligned} \tag{7}$$

where x_i and y_i gives the UAV location, ψ_i is its heading, ψ_i^d is the desired heading, v_i is the ground speed, and k is autopilot gain. We assume the heading rate to be constrained within bounds. That is

$$-\omega_{\max} \leq \dot{\psi}_i \leq \omega_{\max} \tag{8}$$

The UAVs perform a search task to detect targets. When a target is found, a coalition with other agents is formed to prosecute the target. In the next section, the coalition formation process is discussed.

3 Coalition Formation

The heterogeneous UAV swarm has to prosecute the targets by forming coalitions. A coalition is formed based on the target resource requirements for prosecution. A coalition is a temporal team and the agent that initiates a coalition formation request is the *coalition leader* while the agents which form the rest of the coalition are called as *coalition members*. The coalition leader may not necessarily be a part of the coalition because (a) it does not possess the required resources, or (b) the rest of the coalition can perform the task without its presence, or (c) the coalition leader is committed to be a part of another coalition. The role of the coalition leader is to determine a coalition and provide the simultaneous strike information to the coalition members. Once the coalition is formed and the leader broadcasts the coalition information, its role ceases.

In the search region, many agents can detect targets and assume the role of a coalition leader for their detected targets. When a coalition leader announces his target information, all the agents that have not been assigned to any target and who have the desired resources will respond. Other coalition leaders may also respond to the coalition formation requests. When an agent detects multiple targets simultaneously,

it randomly chooses one of the targets and becomes coalition leader to form a coalition to prosecute that target.

Since, the number of available agents are limited, there can be situations where none of the coalition leaders are able to form coalitions causing deadlocks. Deadlock is a situation where multiple coalition leaders attempt to form coalitions and none of the coalition leaders is able to form a coalition. This situation can occur since the total available UAV resources (sums of all the UAV resources) are not properly distributed between the coalition leaders. The deadlock will eventually be solved with time as the agents are in motion in different directions and hence some of the agents will lose their coalition leader role. However, this situation causes delay in forming the coalitions and hence the objective of minimize the coalition prosecution time may not achieved efficiently. Note that the deadlock is different from a situation where the total available agent resources are not sufficient to prosecute a target, in which case, the target will never be prosecuted.

To accomplish mission efficiently, the agents use unique token numbers which are assigned to them prior to the commencement of the search mission. The token mechanism determines the preference with which agents respond to simultaneous multiple coalition formation requests. Thus, when an agent receives multiple coalition formation requests, then, from the broadcast information, the agent can determine which coalition leader has the highest token number. The agent will respond to that highest token number coalition leader. The deadlock breaking mechanism using token numbers is implicit, in the sense; there is no need of additional information exchange between the agents to decide to which coalition leader they should propose.

Other deadlock avoiding mechanisms can be used. For example, consider the ETA of a potential coalition member to arrive at a target location published by a coalition leader as the deadlock avoidance mechanism. Assume an agent A_k receives requests from various potential coalition leaders. Then the agent will determine ETA for each of the published targets and selects a coalition leader according to its minimum ETA.

Although using ETA is a simple metric it may not avoid deadlocks consistently. For example, consider the situation shown in Fig. 1, where agents A_1 and A_2 detect targets T_1 and T_2 respectively and need to form coalitions. Let the target resources be $\mathcal{R}^{T_1} = (2, 3, 4)$, and $\mathcal{R}^{T_1} = (3, 1, 2)$, and agent resources be $\mathcal{R}^{A_1} = (1, 2, 1)$, $\mathcal{R}^{A_2} =$

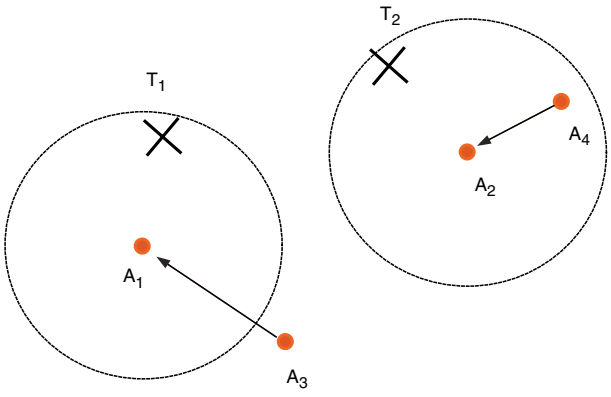


Fig. 1 Situation where multiple agents detect multiple targets

$(2, 1, 0)$, and $\mathcal{R}^{A_3} = (1, 1, 2)$, $\mathcal{R}^{A_4} = (1, 1, 1)$. When the coalition formation request is broadcast by A_1 and A_2 for T_1 and T_2 respectively, then agent A_1 and A_3 will respond to A_1 proposal as their ETA to T_1 is smaller than ETA to T_2. The agents A_4 and A_2 send their bid to A_2, because their ETA to T_2 is smaller than to T_1. In this case, the coalition resources of A_1 for target T_1 are $\mathcal{R}_1^{C^1} = (2, 3, 3)$, while that of A_2 for T_2 are $\mathcal{R}_2^{C^2} = (3, 2, 1)$. We can see that $\mathcal{R}_1^{C^1}$ and $\mathcal{R}_2^{C^2}$ do not meet the $\mathcal{R}^{T_1}$ and $\mathcal{R}^{T_2}$ resource requirements and therefore no coalitions are formed causing a deadlock. This situation may prevail for some more time until the geometry of the agent locations change and one of the coalition leaders finds sufficient coalition resources to assign a coalition.

For the above example if we use the token mechanism then the agents will respond to A_2 first as A_2 has higher token number than A_1. The agents A_1, A_2, A_3, and A_4 will respond to the request and a coalition constituting A_2 and A_3 is formed. After A_2 determines its coalition, A_1 will broadcast again and A_4 will send its information as a response to the broadcast of A_1. But the cumulative resources of A_1 and A_4 do not satisfy the target requirement for T_1, hence a coalition is not formed. Using this scheme, we can see that at least one coalition has been formed to execute the task as compared to no coalitions. As long as the sum of resources of the members intending to be a part of the coalition is greater than or equal to the resources of the target, token mechanism guarantees that at least one coalition is formed. For simulations in this paper, we use the token mechanism.

Once an agent detects a target, it has to form a coalition that depends on the current state of the agent. Figure 2 shows two different states (S_1 and S_2) and how the agent makes a coalition. The sequence of actions carried out during these states are described below.

3.1 Coalition Leader Has All the Required Capabilities (S1)

Agent A_i detects target T_j that requires $\mathcal{R}^{T_j}$ resources. If $R_p^{A_i} \geq R_p^{T_j}$, $\forall\, p = 1, \ldots, m$, and if A_i is not already a part of another coalition, then A_i would attack target T_j without requesting a coalition with other UAVs. This is a special case of coalition formation where only one UAV comprises the coalition. The UAV determines the route to travel using Dubins curves [21], then proceeds towards the target as shown by the state S1 in Fig. 2. In the above example, if A_1 had all of the required resources, then it will prosecute the target without sending a broadcast for a coalition.

3.2 Coalition Leader Has Partial Resources or No Resources (S2)

This state occurs when an agent A_i detects target T_j that requires $\mathcal{R}^{T_j}$ resources, but A_i has insufficient resources. In this case, A_i assumes the coalition leader role and broadcasts the information about the target (i.e, its location and required resources) to the other UAVs. The agents that have at least one type of the required resource will send their cost (ETA) to arrive at the target and its resource vector. The coalition leader considers all the responses and determines if a coalition can be formed or not. If a coalition cannot be formed then it sends a discard coalition broadcast as shown in Fig. 2, otherwise, it will form a coalition and broadcast the coalition information with the time to prosecute the target. The selected potential members will re-plan their paths using the developed simultaneous strike mechanism (described in Section 3.3)

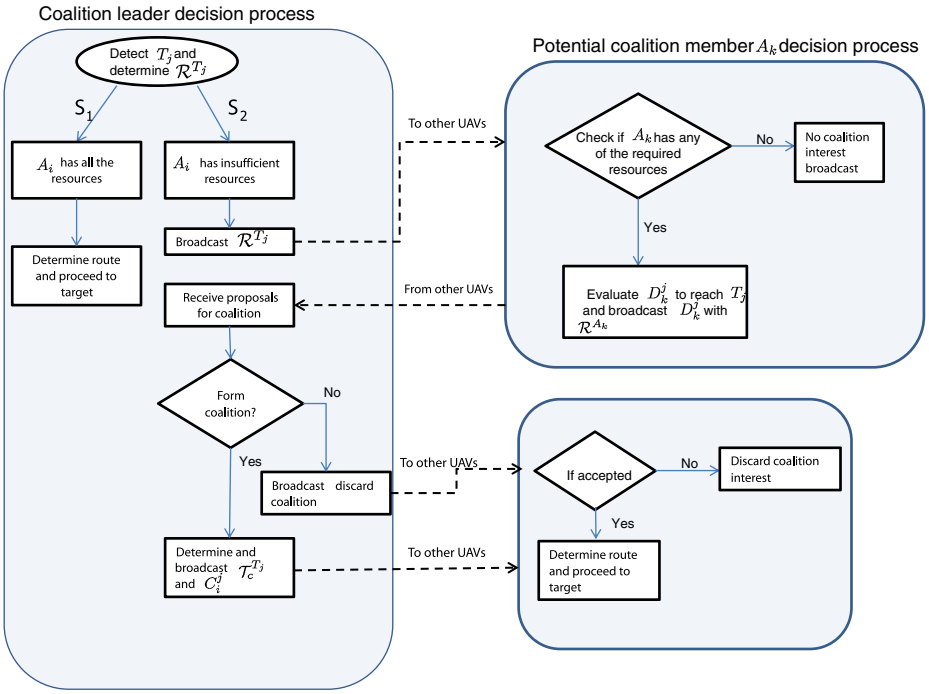


Fig. 2 Sequence of events during coalition formation for the coalition leader and the potential coalition members

to prosecute the target. The rejected agents will discard their potential interest in the coalition and continue to perform their search task. During this process the potential members who announced their availability for coalition will be in a wait state to receive information from the coalition leader. The complete sequence of actions that happen during the coalition formation for a target by the coalition leader is shown in Fig. 2.

The coalition leader has to solve the optimization problem given in Eqs. 5 and 6 to determine the coalition members. Since solving the optimization problem is computationally intensive we developed two coalition formation algorithms: (i) Polynomial time coalition (which is sub-optimal) and (ii) Optimal coalition. Both the algorithms use a two-stage mechanism to produce solutions that have low computational complexity. These two algorithms are similar to what we proposed in [30].

3.2.1 Polynomial Time Coalition Formation Algorithm (PTCFA)

Determining the minimum time and the smallest coalition that would successfully prosecute the target can be accomplished in two stages. In the first stage, we determine the set of all UAVs that can achieve the minimum time requirement and then we prune this set to achieve the minimum member coalition in the second stage. The process to achieve this task is shown by Algorithms 1 and 2. We assume that agent A_i is the coalition leader and it has detected target T_j that requires $\mathcal{R}^{T_j}$ resources.

Algorithm 1 First stage of the PTCFA

1: *Initialize:*
2: $C_j^i = [\,]$ and $\mathcal{R}_j^{C^i} = [\,]$
3: agents_responded = $\{A_1, A_2, \cdots, A_N\}$
4: ETAs = $\{D_1^j, \cdots, D_N^j\}$
5: $\mathcal{D}_c = []$;
6: *Initialize:*
7: \\ sorting of the responses by their ETAs in ascending order.
8: $[\mathcal{D}_u,\ \mathcal{D}_a] = Sort$ (ETAs); % $\mathcal{D}_u \leftarrow$ sorted ETAs, $\mathcal{D}_a \leftarrow$ corresponding agent index of $\mathcal{D}_u$
9: **for** $k = 1$ to |agents_responded| **do**
10: $A_q = \mathcal{D}_a(k)$;
11: $C_j^i \leftarrow$ append A_q to C_j^i
12: $\mathcal{R}_j^{C^i} \leftarrow \mathcal{R}_j^{C^i} + \mathcal{R}^{A_q}$
13: $\mathcal{D}_c \leftarrow$ append $\mathcal{D}_u(k)$;
14: **if** $R_{jp}^{C^i} >= R_p^{T_j}$, for all p **then**
15: Return$\left(C_j^i, \mathcal{D}_c, \mathcal{R}_j^{C^i}\right)$
16: BREAK
17: **else**
18: CONTINUE
19: **end if**
20: **end for**
21: Return('No Coalition')

The Algorithm 1 begins with initializing the coalition set and the coalition resources set to empty sets (line 2). First the coalition leader sorts the responses in the ascending order of cost (line 9). We take one agent (A_q) at a time (line 11), include A_q in the coalition C_j^i (line 12), update the coalition resources set $\mathcal{R}_j^{C^i}$ (line 13) and the coalition time set $\mathcal{D}_c$. Then check if the target resource constraint is met (line 15). If the constraint is not met, then the process of including the next agent and

Algorithm 2 Second stage of the PTCFA

1: $\hat{C}_j^i = C_j^i$;
2: **for** $k = 1 : |C_j^i|$ **do**
3: $A_q = C_j^i(k)$;
4: $\hat{\mathcal{R}}_j^{C^i} = \mathcal{R}_j^{C^i} - \mathcal{R}_j^{C^q}$
5: **if** $R_{jp}^{C^i} >= R_{jp}^{T},\ \forall\ p$ **then**
6: $\hat{C}_j^i \leftarrow$ remove A_q from $\hat{C}_j^i$
7: $\mathcal{R}_j^{C^i} = \hat{\mathcal{R}}_j^{C^i}$
8: **end if**
9: **end for**
10: $C_j^i = \hat{C}_j^i$;

its resources and verifying the resource constraint continues until the target resource constraint is met.

Once the resource constraint is met, the algorithm terminates. If the target resource constraint cannot be met despite adding all the agents in list $\mathcal{D}_a$, then a successful coalition cannot be formed.

To illustrate Algorithm 1, we give a hypothetical example where agent A_1 detects a target that requires only two types of resources. The target resource requirement is $\mathcal{R}^{T_1} = (5, 3)$, with $R_1^{T_1} = 5$ and $R_2^{T_1} = 3$. Suppose that the UAVs that responded with their resources and cost to the coalition leader are given in the Table 1. As per stage 1 of the algorithm, the possible coalition members are sorted in ascending order based on cost. The new list is given in Table 2. The ordered set of coalition members is $\mathcal{D}_a = (A_2, A_3, A_6, A_1, A_4, A_5)$. The coalition is formed by recruiting members from this ordered set starting with the first member. The coalition formation process progresses in the following way. Initially A_2, the first member in the ordered set is selected into the coalition making the coalition $C_1^1 = \{A_2\}$ with $\mathcal{R}_1^{C^1} = (1, 3)$. Since $\mathcal{R}_1^{C^1}$ is not sufficient to prosecute the target, the next agent, A_3, is added to C_1^1 forming $C_1^1 = (A_2, A_3)$ with combined resources of $\mathcal{R}_1^{C^1} = (2, 4)$. As $\mathcal{R}_1^{C^1}$ still does not suffice, the next agent A_6 is added to C_1^1 which becomes (A_2, A_3, A_6) and the coalitional resource set becomes $\mathcal{R}_1^{C^1} = (2, 6)$. The coalition resources are still not sufficient, hence the next agent A_1 is included in C_1^1 resulting in $\mathcal{R}_1^{C^1} = (4, 7)$. The $R_1^{C^1}$ is still short of the requirement and A_4 is recruited to the coalition resulting in $C_1^1 = (A_2, A_3, A_6, A_1, A_4)$ and $\mathcal{R}_1^{C^1} = (6, 7)$, which satisfies the target resource requirement. The first stage of the algorithm terminates once it determines the coalition that satisfies the target resource constraint. The prosecute time for simultaneous rendezvous is determined by the coalition member whose ETA is highest. In the present case, the ETA of A_4 is the highest with 172 *s*. The procedure in Algorithm 1 will ensure a coalition with minimum time to target. This is proved in the following theorem.

Theorem 1 *Algorithm* 1 *generates an optimal minimal time coalition set.*

Proof Let N be the number of agents that responded to a coalition formation request to prosecute target T_j that requires resources R^{T_j}. The set of potential coalition members is given by $\mathcal{C} = \{A_1, \ldots, A_N\}$ with corresponding costs (ETA to target) as $\mathcal{D}_i^j = \{D_1^j, \ldots, D_N^j\}$. Assume that $\sum_{i=1}^{N} \mathcal{R}^{A_i} \geq \mathcal{R}^{T_j}$. Algorithm 1 orders the set $\mathcal{C}$ to form $\mathcal{C}_o = (A_{i_1}, \ldots, A_{i_N})$ where $i_1, \ldots, i_N$ is a permutation of $1, \ldots, N$

Table 1 List of agents who responded to the coalition formation request in the example, their available resources and costs to target

UAV	Resources	Cost (s)
A_1	2, 1	123
A_2	1, 3	47
A_3	1, 1	63
A_4	2, 0	172
A_5	3, 2	207
A_6	0, 2	96

Table 2 List of agents in the example, after sorting

UAV	Resources	Cost (s)
A_2	1, 3	47
A_3	1, 1	63
A_6	0, 2	96
A_1	2, 1	123
A_4	2, 0	172
A_5	3, 2	207

such that the corresponding ordered cost $\mathcal{D}_o^j = \left(D_{i_1}^j, \ldots, D_{i_N}^j\right)$ has the property that $D_{i_k}^j \leq D_{i_{k+1}}^j, \forall i \in \{1, \ldots, N-1\}$. For a given $r \in \{1, \ldots, N\}$, let $\mathcal{C}_o^r = \{A_{i_1}, \ldots, A_{i_r}\}$. The target prosecution time for $\mathcal{C}_o^r$ is D_{i_r}. The algorithm finds the least r members such that $\sum_{k=1}^{r} \mathcal{R}^{A_{i_k}} \geq \mathcal{R}^{T_j}$. Let v be the least integer in $\{1, \ldots, N\}$ with the above property. Then $\mathcal{C}_o^v$ is the coalition formed by Algorithm 1 and the claim is that this is a coalition that prosecutes the target T_j is minimum possible time. Clearly, $\mathcal{C}_o^v$ is a feasible coalition as the combined resources of the coalition are enough to prosecute the target. Since the elements of $\mathcal{D}_o^j$ are ordered in ascending order, the addition of an agent A_{i_k} with $k > v$ will not decrease coalition time. Similarly, removal of any agent from $\mathcal{C}_o^v$ other than A_{i_v} will not decrease the coalition time which is $D_{i_v}^j$. Whereas, removal of A_{i_v} will result in an infeasible coalition as v is the least integer such that $\mathcal{C}_o^v$ satisfies the resource requirement. Thus $\mathcal{C}_o^v$ is a feasible coalition with minimum time to prosecute target T_j. □

Once the minimum time coalition is formed by Algorithm 1, we need to prune those members whose resources are not required to form a minimum member coalition. This process is carried out by using Algorithm 2. In the second stage, we check if the resources of each agent $A_q \in C_j^i$ in the coalition formed in stage 1 are required for the coalition or not by removing its resources from $\mathcal{R}_j^{C^i}$ (line 5). If resources of A_q are not required (line 6), then the agent A_q is removed from C_j^i (line 7) and its resources are deducted from the $\mathcal{R}_j^{C^i}$ (line 8). This process is carried out for all the agents $A_q \in C_j^i$. The process of the second stage is described in Algorithm 2.

We continue the example given above to illustrate the second stage of the coalition formation algorithm. Let $\hat{C}_1^1 = C_1^1$ be a temporary coalition with $\mathcal{R}_1^{C^1}$ as its resources. The first agent in C_1^1 is A_2, therefore first we remove its resources from $\mathcal{R}_1^{C^1}$ that results in $\hat{\mathcal{R}}_1^{C^1} = (5, 4)$. As the target requirement is $\mathcal{R}_1^T = (5, 3)$, the condition $\hat{\mathcal{R}}_1^{C^1} \geq \mathcal{R}_1^T$ is met, hence the agent A_2 and its resources are removed. This results in $\hat{C}_1^1 = \{A_3, A_6, A_1, A_4\}$ and $\mathcal{R}_1^{C^1} = \{5, 4\}$. The next agent in C_1^1 is A_3, its resources are removed from $\mathcal{R}_1^{C^1}$ resulting in $\hat{\mathcal{R}}_1^{C^1} = (4, 3)$. Removing agent A_3 from $\hat{C}_1^1$ does not meet the target resource constraint, hence A_3 and its resources are restored in the coalition. If we remove any of the remaining agents in C_1^1, the target requirement will not be met. Hence the coalition after second stage is $C_1^1 = (A_3, A_6, A_1, A_4)$ with resources $\mathcal{R}_1^{C^1} = (5, 4)$. Determining the minimum member coalition set as above involves less computational complexity but can produce sub-optimal solutions as all possible sub-coalitions and their resource contributions are not taken into account.

3.2.2 Optimal Coalition Formation Algorithm (OCFA)

To determine the coalition which has an optimal size, we formulate an integer programming problem for the second stage of the two-stage algorithm. The formulation of the problem is given in Algorithm 3.

Algorithm 3 Optimal coalition formation algorithm

1: Stage 1:
2: Use Algorithm 1 to determine $\mathcal{D}_c$

3: Stage 2:
4: Solve:

$$Objective\ Function\ :\ \min_{\mathcal{D}'_c \subseteq \mathcal{D}_c} |\mathcal{D}'_c| \tag{9}$$

$$\text{subject to} \sum_{A_k \in \mathcal{D}'_c} R_p^{A_k} \geq R_p^{T_j},\ \text{for all } p = 1, \ldots, m \tag{10}$$

Algorithm 1 provides optimal minimum time and the Algorithm 3 generates an optimal minimum number of agents, hence the two-stage algorithm generates an optimal minimum time and minimum size coalition. We continue the example given above to describe the functioning of the second stage of the optimal coalition formation algorithm. The optimal coalition after solving the integer programming problem yields $C_1^1 = (A_2, A_1, A_4)$ with $\mathcal{R}_1^{C^1} = (5, 4)$.

From the coalitions obtained using PTCFA and OCFA, we can see that the coalition formed using OCFA has the smaller coalition with three members and it is the optimal solution for a single target. While the PTCFA is a sub-optimal solution.

3.2.3 Complexity Analysis

Now we analyze the complexity of the PTCFA and OCFA algorithms. The PTCFA produces a sub-optimal solution that has polynomial time complexity. Hence, it can be used for real-time applications.

Theorem 2 *The computational complexity of the PTCFA composing of Algorithms* 1 *and* 2 *is* $O(N(\log N + 2m))$.

Proof Assume that all N agents respond to the coalition request and the target T_j requires m types of resources to prosecute. The first stage (using Algorithm 1), which includes the sorting of proposals requires $O(N \log N)$ computations. The step 16 or 22 needs Nm computations, hence the computational complexity of the first stage is $O(N \log N) + O(Nm)$. In the second stage (using Algorithm 2) we need Nm computations for step 6. Therefore, the algorithm complexity of PTCFA is $O(N \log N) + O(2Nm)$. □

The computational complexity for the optimal coalition algorithm has a polynomial time complexity for the first stage. But we use an integer programming

technique in the second stage. Although solving an integer programming problem is *NP*-hard, there are pseudo-polynomial algorithms to generate optimal solutions [29]. To solve the integer programming problem in the second stage, we used the *bintprog* command in MATLAB that in turn uses a branch and bound technique to compute the solution.

When N is very large, the computational time for optimal size coalition depends on two factors: the number of agents and the quantity of their resources. During the initial phase of the mission, the resources of the UAVs are full and hence the coalition leader may receive a higher number of proposals. However, since the UAVs are full of resources the coalition selected after stage 1 ($\mathcal{D}_c$) will have lower number of UAVs. As the minimum member coalition is to be found from a subset of $\mathcal{D}_c$ which is small, the computational time will be small. The case is reversed during the final stages where the UAVs may have fewer resources and the coalition size is necessarily larger to meet the target resource requirements. As the coalition size becomes large the computational time increases. The above analysis indicates that the computational time for the optimal member coalition formation mainly depends on the distribution of the UAV resources.

3.3 Simultaneous Strike

The coalition leader A_i determines the latest arrival time $\mathcal{T}_c^{T_j}$ for the coalition and provides this information to the accepted coalition members. The coalition members need to adjust their paths such that the time to arrive at the target is equal to $\mathcal{T}_c^{T_j}$. There are several methods that can be used to achieve simultaneous strike by solving the rendezvous problem [15–20]. However, these methods require continuous communication to achieve rendezvous resulting in an increase in communication cost. Moreover, the kind of operations that we consider are carried out in hostile territory where the communication needs should be minimum. We developed a different approach where there is no communication between the coalition members to achieve simultaneous strike. In this approach, the agents find the radius $\underline{r}$ (bounded below by the minimum turning radius) of the Dubins curve to be followed by the agent such that the ETA of the agent is equal to $\mathcal{T}_c^{T_j}$. To execute this method, the coalition members need only the information of $\mathcal{T}_c^{T_j}$ from the coalition leader.

Assume that agent A_i is the coalition leader for target T_j forming a coalition C_j^i with latest arrival time of $\mathcal{T}_c^{T_j}$. Each member $A_{i_k} \in C_j^i, i_k \in \{1, \ldots, N\}, k = 1, \ldots, |C_j^i|$ has to adjust its path length such that D_{i_k} is equal to $\mathcal{T}_c^{T_j}$. In order to achieve that, the agents need to find their turning radius $\underline{r}_k$ such that $D_k^j = \mathcal{T}_c^{T_j}$. Since, $\underline{r}_k$ cannot be calculated using a closed form solution, we calculate $\underline{r}_k$ iteratively until the condition $D_k^j = \mathcal{T}_c^{T_j}$ is satisfied.

Cost: The selection of the coalition members in the first stage of PTCFA and OCFA is based on the ETA of the agent from the target. Since the agents have kinematic constraints, they need to use Dubins curves [21] to compute ETA.

Given an agent position with its heading, two Dubins paths can be determined to the target: (i) Dubins shortest path and (ii) Dubins longest path, as shown in Fig. 3a. The time taken to follow either of these two paths can be used by the agents as *cost*. However, the selection of (i) or (ii) depends on whether they can provide cost

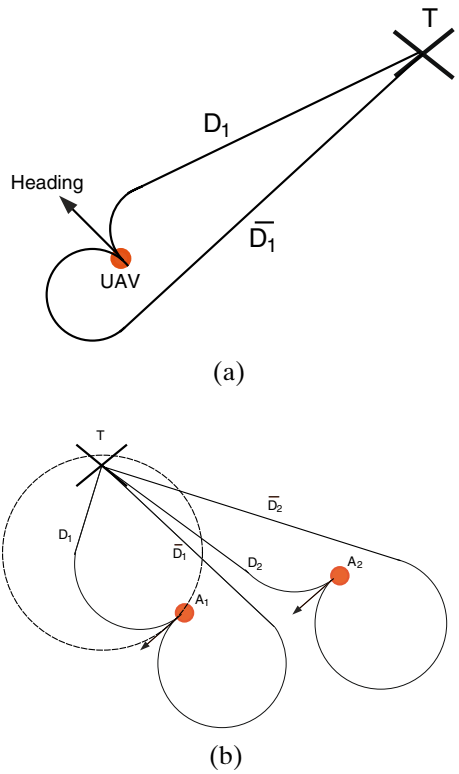


Fig. 3 **a** Dubins curves, where D_1 is the Dubins shortest path, while $\bar{D}_1$ is the Dubins longest path. **b** An example of selecting Dubins path

(path length or ETA) as a continuous function of radius. This is important for the simultaneous attack condition.

Consider the example as shown in Fig. 3b, where agent A_1 is the coalition leader and A_2 is a member of the coalition. If we choose the ETA given by the Dubins shortest path, then accordingly A_1 has to adjust its path length such that D_1 is equal to D_2, since $\mathcal{T}_c^T = D_2$. But, it is not possible for A_1 to determine a path if the radius of the circle encircles the target, as shown by the dotted circle in Fig. 3b. In such a case, A_1 might want to choose $\bar{D}_1$, but then it may be that $\bar{D}_1 > D_2$ and therefore A_1 cannot travel along $\bar{D}_1$. This occurs because there is a discontinuity in the achievable ETA when one uses Dubins shortest path. To eliminate this discontinuity, we always use the Dubins longest path as the metric for cost. In this case, A_1 can increase its radius to match the ETA of A_2 for any $\bar{D}_2 > D_1$. The simultaneous strike constraint for a coalition C_j^i is obtained by the individual agents modifying the radius of their Dubins longest path to the target to match $\mathcal{T}_c^{T_j}$.

To calculate the ETA of an agent using the Dubins longest path from its current position to a target, we need to consider two cases as shown in Fig. 4. Case (a): the target is on the right hand side of the UAV and the UAV has to turn counter-clockwise to follow the Dubins path to the target. Case (b): the target is on the left hand side of the UAV and the UAV has to make a clockwise turn to reach the target via Dubins longest path.

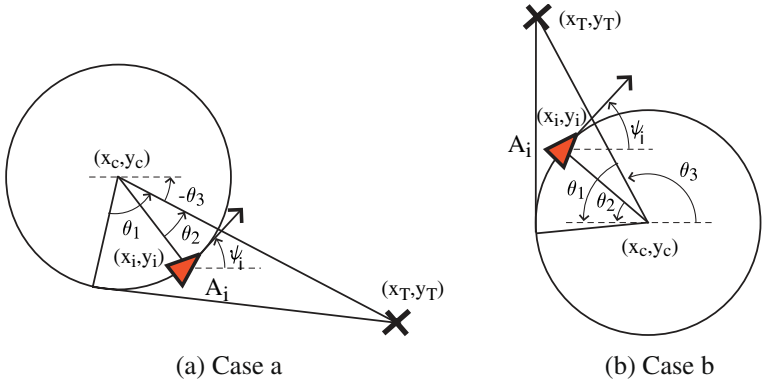


Fig. 4 Calculation of Dubins distance

Let (x_i, y_i) be the location of an agent and ψ be its heading. Let (x_T, y_T) be the target location. Let (x_c, y_c) be the center of the circle tangent to the UAV's velocity and drawn away from the target (in the direction of turn). For case (a), this is given as

$$x_c = x_i - r\sin\psi \tag{11}$$

$$y_c = y_i + r\cos\psi \tag{12}$$

and for case (b), this is

$$x_c = x_i + r\sin\psi \tag{13}$$

$$y_c = y_i - r\cos\psi \tag{14}$$

where r is the radius of the circle along which the agent makes a turn.

In both the cases, we find angles θ_1, θ_2 and θ_3 as follows.

$$\theta_1 = \arccos\left(\frac{\sqrt{(x_T - x_c)^2 + (y_T - y_c)^2}}{r}\right) \tag{15}$$

$$\theta_2 = \frac{\pi}{2} - \psi \tag{16}$$

$$\theta_3 = \arctan\left(\frac{y_T - y_c}{x_T - x_c}\right) \tag{17}$$

In case (a), the ETA is given as

$$D = (r\{2\pi - [\theta_1 - ((\pi - \theta_3) - \theta_2)]\} + r\tan\theta_1)/v_i \tag{18}$$

and in case (b), it is given as

$$D = (r\{2\pi - [\theta_1 - (\theta_2 + \theta_3)]\} + r\tan\theta_1)/v_i \tag{19}$$

where v_i is the ground speed of the agent.

Note that each agent's Dubins longest path for the simultaneous strike constraint can be found by adjusting the corresponding radius. One may argue that instead of using Dubins longest path, it is better to use Dubins shortest path by increasing the

radius until the radius encircles the target and when the shortest path encircles the target, then use longest path. Alternatively, another approach could be to move away from target and determine if the Dubins shortest path can allow you to reach the target. Although these approaches are intuitively appealing, there is no guarantee that they will produce the required simultaneous strike trajectories. Since, the UAV is in motion, the solution should be quick. Any of the above alternatives, including the Dubins longest path approach, are concrete in providing the solution. Hence, to have a generic model with a single rule we adopted the Dubins longest path. In the worst case, the time need to prosecute target T_j increases by $2\pi r_{\min}/V$ s as compared to the time taken using Dubins shortest path.

Once the coalition is formed, the agents travel along the new derived path. During this process, the coalition members may detect other targets. In that case, the detecting member will become a coalition leader for that target and carry out the coalition generation process. However, this agent will not participate as a member of the coalition, since it is already a part of another coalition. From the computational analysis described in the previous section, the two-stage algorithm has low computational complexity and require only three broadcasts to make a coalition. Hence, they can afford to be the coalition leaders for the detected targets instead of allowing some other agent to become a coalition leader.

The algorithms developed in this paper are used to form coalitions and prosecute targets as and when they are detected. Algorithm 3 provides an optimal solution to prosecute a single target. But, it is necessary to know the deviation of the proposed solution in a multiple target scenario from that of the global optimal solution where the information about all targets are known *a priori*. For this purpose, we formulate the problem as a global static optimization problem and solve it. In the next section, we will discuss the solution concepts for this combinatorial optimization problem.

4 Combinatorial Optimization Problem

The solution to the combinatorial optimization problem is NP-hard, in part because it is very difficult to capture the dynamics of the resources depletion after prosecuting a target. Therefore we will solve the optimization problem using the *Particle Swarm Optimization* (PSO) technique. The PSO allows us to model the optimization problem taking the resource dynamics into account. Also, PSO is faster compared to genetic algorithms [26]. Due to these features we adopted PSO to solve the optimization problem with known target and UAV positions and resources.

4.1 Overview of Particle Swarm Optimization

PSO is a population based stochastic optimization technique developed by Eberhart and Kennedy [23, 24]. The key idea behind the algorithm is to simulate the social behavior of bird flocks, fish schools, etc. Each particle in a swarm is a potential solution in the search space. The particle adjusts its velocity according to its own flying experiences and its flock's experiences. The PSO technique is similar to the evolutionary computation techniques in [26], however, in PSO each particle can adapt its velocity to move in the search space and has memory of its best position.

If we assume the optimization problem to be of Q-dimension, then each particle in the swarm S can be represented as $X_l = (x_{l1}, x_{l2}, \ldots, x_{lQ}), l = 1, \ldots, S$. The best previous position attained by the particle is represented as $P_l = (p_{l1}, p_{l2}, \ldots, p_{lQ})$, and the velocity of the particle is $V_l = (v_{l1}, v_{l2}, \ldots, v_{lQ})$. The best global position achieved by the swarm is represented as $P_g = (p_{g1}, p_{g2}, \ldots, p_{gQ})$, and the iteration number is represented as superscript k. The particles in the swarm are updated according to the following equations

$$v_{ld}^{k+1} = \chi\left(w v_{ld}^{k} + c_1 r_1\left(p_{ld}^{k} - x_{ld}^{k}\right) + c_2 r_2\left(p_{gd}^{k} - x_{ld}^{k}\right)\right), \tag{20}$$

$$x_{ld}^{k+1} = x_{ld}^{k} + v_{ld}^{k+1} \tag{21}$$

where $d = 1, \ldots, Q$, r_1 and r_2 are uniformly distributed random numbers between $[0,\ 1]$, c_1 and c_2 are positive constants representing cognitive and social parameters, w is the inertia weight, and χ is the constriction factor. The role of inertia weight w is to create a balance between global and local explorations. Initially, it is necessary to explore the search space and then reduce w as the solution reaches the optimum value [25]. The constants c_1 and c_2 aid in convergence of the solution. The random parameters r_1 and r_2 are used to maintain the diversity of the swarm population, while the constriction factor controls the effect of velocity on the particles.

The update equations given in Eqs. 20 and 21 are modified from the original equations developed by Eberhart and Kennedy [23], which did not have the inertia and constriction factors. In the next section we will apply the PSO algorithm to determine coalitions for all the targets and the sequence in which the targets are prosecuted.

4.2 Solution to the Static Optimization Problem Using PSO

The value of the particle in each dimension can, in general, be a decimal value, but for the target assignment problem we need it to be an integer. The usual method of using PSO for integer programming optimization methods is to round the value of the particle [27, 28]; we will be using the same scheme in evaluating the objective function.

The notation of the particle X_l is modified to suit our problem. We assume that N UAVs and M targets are present in the search space. The l^{th} particle at the k^{th} iteration is $X_l^k = \left(x_{11}^{l,k}, \ldots, x_{1M}^{l,k}, x_{21}^{l,k}, x_{22}^{l,k}, \ldots, x_{2M}^{l,k}, \ldots, x_{N1}^{l,k}, x_{N2}^{l,k}, \ldots, x_{NM}^{l,k}\right)$, where $x_{ij}^{l,k} \in \{0, \ldots, M\}$ and indicates that agent A_i will prosecute target $x_{ij}^{l,k}$ in j^{th} sequence. A value of 0 for $x_{ij}^{l,k}$ represents a search. The dimension of the particle is NM. The first M dimensions of the particle present the order in which the first agent would execute different targets. The dimensions $M+1$ to $2M$ present the order in which the second agent will attack targets, and so on. Thus we have N agents' preferences of the targets and using these preferences the PSO evaluates the cost of the mission using Algorithm 4.

First the PSO algorithm parameters are initialized in lines 1 and 2 of Algorithm 4. For each iteration, all the particles are evaluated using *evaluate_swarm* function. Before evaluating the particle in this function we check if the boundary conditions are met or not. If they are not met then the value of the particle is set to infinity (line 18). The value of the particle is also set to infinity if the solution does not exists (line 34), otherwise the particle is evaluated (lines 20–39). The cost of the particle in line 29, is the time taken by a coalition to prosecute that target. Once all the particles

Algorithm 4 Global optimal solution using PSO

```
1: Initialize: S, χ, ω, c1, c2, number_of_iterations.
2: bestParticleValue = ∞; bestParticle = []
3: for Iter = 1 to number_of_iterations do
4:    Pval ← evaluate_swarm(S); % Evaluate each particle and determine its value
5:    [minParticleValue, minParticleIndex] ← min(Pval)
6:    if minParticleValue ≤ bestParticleValue then
7:       bestParticleValue ← minParticleValue;
8:       bestParticle← S(minParticleIndex);
9:    end if
10:   check termination conditions
11:   update S using Eqs. 20 and 21
12: end for
13: Function cost = evaluate_swarm(S)
14: for l = 1 : S do
15:   cflag ← check if the value of each dimension of the particle is between [0, M].
16:   targetsAssigned ← [];
17:   if cflag == 0 then
18:      cost(l) ← ∞
19:   else
20:      cost(l) ← 0;
21:      for j = 1 : M do
22:         currPref ← S(l, j : M : NM)
23:         while currPref ≠ ∅ do
24:            target ← currPref(1)
25:            if find(target == targetsAssigned)==∅ then
26:               agentsPrefTarget ← find(currPref == target);
27:               cr ← Check the sum of resources of agentsPrefTarget
28:               if cr ≥ R_j^T then
29:                  cost(l) ← cost(l) + max {dubinDistance(agentsPrefTarget)}
30:                  Reduce the resources of the agents in agentsPrefTarget
31:                  targetsAssigned ← [targetsAssigned target]
32:                  Record agent to target mapping
33:               else
34:                  cost(l) ← ∞
35:               end if
36:            end if
37:            currPref = setdiff(currPref, target)
38:         end while
39:      end for
40:   end if
41: end for
```

are evaluated, we determine the l^{th} particle that has minimum value (minParticleValue) and the particle in the swarm S corresponding to the minimum value (minParticleIndex, line 5). If minParticleValue is less than the previous best value then the bestParticleValue and bestParticle variables are re-assigned (lines 6–9). Then we

check for termination condition and update the swarm using the PSO equations given in (20) and (21) (lines 10 and 11). This process is carried out for all the iterations (given by number_of_iterations) or until a termination condition is met.

The PSO algorithm produces the optimal coalition assignment for all the targets. However, the dimension of the particle depends on the number of agents and targets. With an increase in the number of agents or number of targets or both, the computational time increases as PSO has to search on NM dimensional solution space.

A hypothetical scenario is considered to demonstrate the execution of the PSO algorithm. Consider a scenario with three targets and four UAVs whose initial positions are shown in Fig. 5. The resources of the agents are $\mathcal{R}^{A_1} = (1, 2, 3)$, $\mathcal{R}^{A_2} = (2, 0, 1)$, $\mathcal{R}^{A_3} = (1, 3, 1)$, $\mathcal{R}^{A_4} = (1, 2, 1)$, and the target resource requirements are $\mathcal{R}^{T_1} = (2, 1, 0)$, $\mathcal{R}^{T_2} = (1, 0, 1)$, $\mathcal{R}^{T_3} = (0, 1, 3)$.

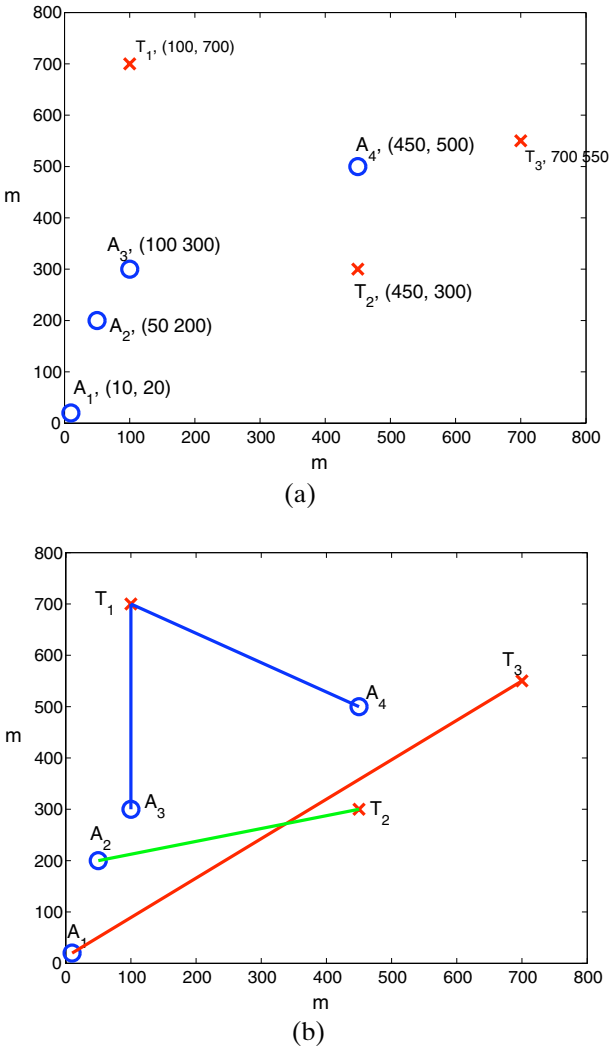


Fig. 5 **a** Initial positions of the UAVs and targets for the example. **b** The target assignment achieved using PSO

The solution generated by the PSO is $X_l = (300020100100)$ after 532 iterations. From the solution, according to step 22 in the Algorithm 3, currPref = [3 0 1 1]. Let the current target be 3 (step 24). Since only one agent has target 3, and it satisfies the resource requirement, A_1 is assigned to T_3 in step 26. The distance of the agent to target using Dubins curves is added to the cost in step 29. The target is removed from the currPref vector which now is [0 1 1]. The next target is 0, which is a search, hence it is not taken into account. Therefore the currPref is [1 1]. The next target is T_1, carrying out procedure similar to T_3 we can see that the resource constraint is satisfied. Hence A_3 and A_4 are assigned to T_1. The cost (latest arrival time of the coalition) to execute the target is added to the cost(l), and the target is removed from the currPref list. Since the currPref list is empty, the next sequence is considered, and now currPref = [0 2 0 0]. Since, target T_2 is the only target present there, it is assigned to A_2. The cost(l) is updated with the new cost of A_2 moving to T_2. The process continues until all the for-loops are completed. Figure 5b shows the assignment of the agents to targets. In the figure we can see that A_1 is assigned to T_3, A_3 and A_4 are assigned to T_1, and A_2 is assigned to T_2. The cost given by the solution X_l is the total time taken to accomplish mission by prosecuting all of the targets.

5 Simulation Results

Monte-Carlo simulations are carried out to evaluate the performances of the single target coalition formation algorithms (PTCFA and OCFA) as well as the global optimal coalitions for all the targets. The complexity of the optimization problem increases with increase in number of agents or number of targets, or both. We will analyze these effect in terms of the time required to complete the mission (mission completion time), time required to form a coalition (coalition time), and the time required to simulate one complete mission with a given target and UAV distribution (simulation time) on both the two-stage algorithms and the static PSO solution.

5.1 Difference in Solutions Obtained Using PSO and Two-Stage Algorithms

We consider a sample mission scenario with four UAVs and four targets and analyze the time taken by each algorithm to form a coalition (coalition time), the time taken to accomplish the mission, and the simulation time (i.e. the time computer takes to simulate a given mission scenario). The region is an area 1,000 × 1,000 m in size, and the initial position of the UAVs and targets are shown in Fig. 6. The velocity of the UAVs is 10 m/s and the sensor range is 300 m. A UAV that goes out of the 1,000 × 1,000 m region of interest during its search takes a minimum radius turn to come back into the region. The available resources of the UAVs are $\mathcal{R}^{A_1} = (2, 3, 4)$, $\mathcal{R}^{A_2} = (2, 1, 3)$, $\mathcal{R}^{A_3} = (3, 2, 4)$, $\mathcal{R}^{A_4} = (2, 2, 0)$ and the required resources of the targets are $\mathcal{R}^{T_1} = (1, 1, 2)$, $\mathcal{R}^{T_2} = (3, 2, 4)$, $\mathcal{R}^{T_3} = (2, 1, 2)$, $\mathcal{R}^{T_4} = (3, 4, 1)$.

5.1.1 Mission Performance Using PTCFA

The agents have to search for targets and when detected need to form coalitions. As shown in Fig. 7a at time $t = 0.6$ s, A_1 detects target T_1 and its resources satisfy the condition $\mathcal{R}^{A_1} \geq \mathcal{R}^{T_1}$. Hence, the agent is in state S1 and it attacks the target without sending a coalition request as described in Section 3.1. The trajectory of A_1

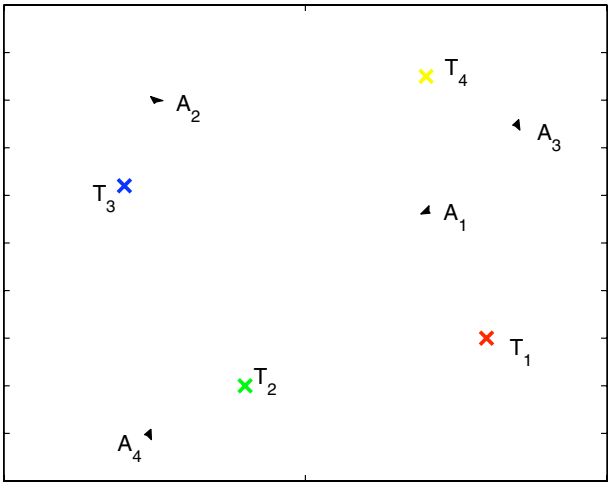


Fig. 6 Sample scenario for determining the difference in mission performance achieved using PTCFA, OCFA and the PSO solution

traveling towards T_1 is shown in Fig. 7a. At time $t = 1$ s, A_3 detects T_4 and forms a coalition with agents A_2 and A_4. These agents adjust their path and travel towards T_4 as shown in Fig. 8a. After prosecuting the assigned targets, the agents continue to search the region. At $t = 147.8$ s, A_3 detects T_3 and a coalition with A_4 is formed. The agents adjust their path length to meet the rendezvous constraint and their desired trajectory is shown in Fig. 9a. On the other hand, T_2 is detected by A_1 at $t = 164$ s and a coalition with A_2 is formed. The route taken by A_1 and A_2 is shown in Fig. 10a. The total mission was accomplished in 263 s. Now, we will repeat the experiment on the mission performance using OCFA, which will demonstrate that the coalition formed using PTCFA is sub-optimal.

5.1.2 Mission Performance Using OCFA

In this simulation, A_1 detects T_1 at $t = 0.6$ s and it assigns itself to T_1. The route followed by A_1 is similar to that using Algorithm 1 as shown in Fig. 7b. At $t = 1$ s,

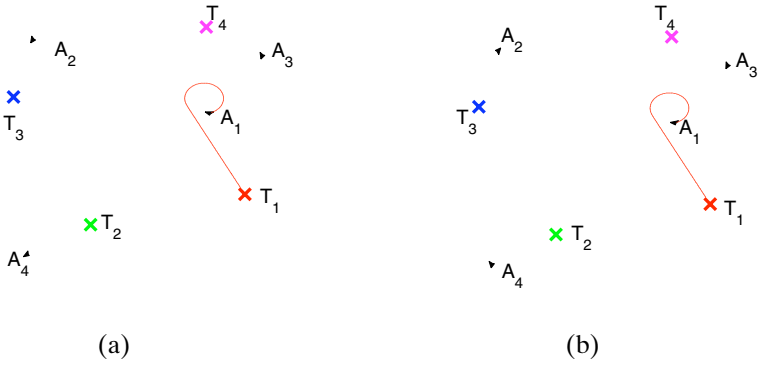


Fig. 7 **a** The trajectory of A_1 attacking T_1 using PTCFA. **b** The trajectory of A_1 attacking T_1 using OCFA

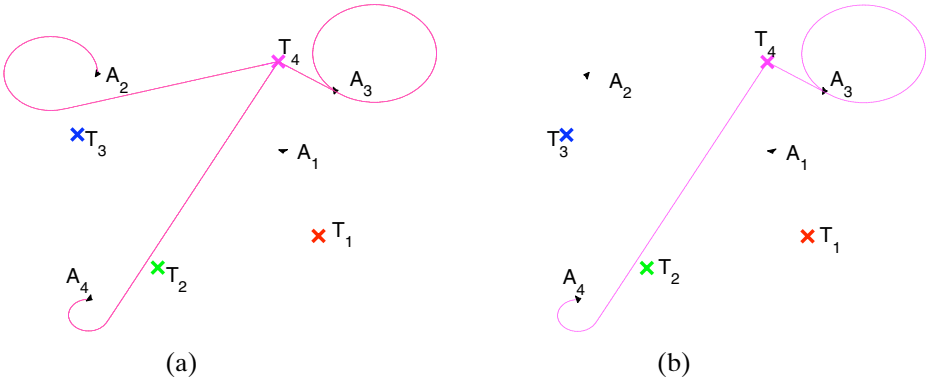


Fig. 8 **a** A_2, A_3, and A_4 are assigned to T_4 PTCFA. **b** A_3 and A_4 assigned to T_4 using OCFA

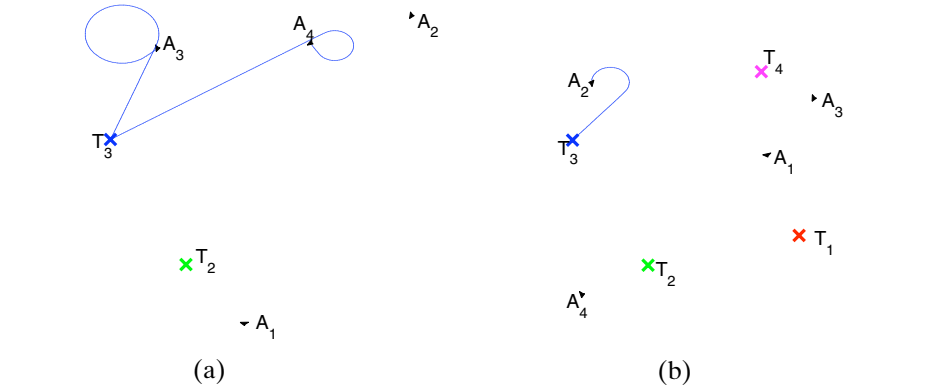


Fig. 9 **a** A_3 and A_4 assigned to T_3 using PTCFA. **b** A_2 assigned to T_3 using OCFA

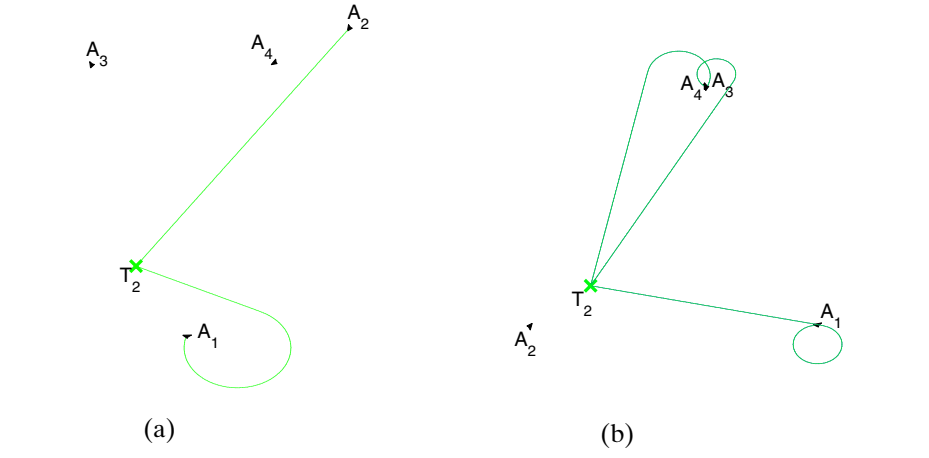


Fig. 10 **a** Trajectories of agents A_1 and A_2 attack target T_2 using PTCFA. **b** Routes of A_1, A_3, and A_4 follow to attack target T_2 using OCFA

A_3 detects T_4, but A_3 generates a coalition with A_4 only and hence the coalition has 2 members, while the coalition generated using PTCFA produced a coalition with three agents. Figure 8b shows the trajectories of A_3 and A_4 towards T_4. The agent A_2 is not assigned to any target and it detects target T_3 at $t = 1.4$ s. Since, its resources are sufficient to prosecute T_3, it generates a single member coalition. The route taken by A_2 to T_3 is shown in Fig. 9b. The final target T_2 is detected by A_1 at time $t = 155$ s and a coalition with A_2, A_3, and A_4 is formed. The route of these agents is shown in Fig. 10b and the mission was accomplished in 213.8 s.

5.1.3 PSO Solution

When the target positions and their resources are known *a priori*, we can solve the combinatorial coalition formation for each agent using PSO. Figure 11 shows the assignment and the routes that the agents take to accomplish the mission. Agent A_1 is initially assigned to T_1 and then assigned to T_4, while A_4 is assigned to T_4 only. So, A_4 takes a path length of A_1 attacking target T_1 plus the Dubins distance of A_1 from T_1 to T_4. The mission can be accomplished in 114.3 s using the assignment determined by the PSO. The optimal solution was obtained with 600 iterations, where each iteration involves solving lines 3–12 in Algorithm 4.

5.1.4 Comparison Analysis

Table 3 summarizes the mission time and simulation time for a mission using PSO solution and the two-stage coalition formation algorithms. The simulations were carried out using MATLAB on a 1.83 GHz, 1 GB RAM machine. The performance of the mission using PTCFA is less than that achieved using OCFA, because the number of agents performing a search task is reduced in PTCFA as the coalitions are not of minimal size and hence the search to detect targets is less efficient. Although the performance of the mission using OCFA is better, the computational time is higher than the PTCFA algorithm.

From the table we can see that the mission can be accomplished earlier using the PSO solution than using any of the two-stage algorithms as the target positions are

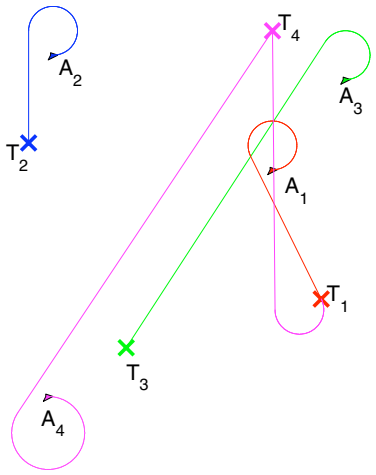


Fig. 11 PSO coalition assignment for all the targets

Table 3 Comparison of the mission with PSO solution and mission with two-stage algorithm

	PSO (s)	PTCFA (s)	OCFA (s)
Mission time	114.3	263	213.8
Simulation time	17.3	3.31	3.83

known *a priori* in this case. However, the simulation time taken to complete the mission using the PSO is much higher than that of two-stage algorithms. In case of PSO solution, the time taken to produce a solution increases exponentially as we increase the number of agents and targets. This effect is studied in the next subsection. Another advantage of using *two-stage* algorithms is that it is decentralized and can take any uncertainties in the environment into account, that may include incomplete knowledge of the resources and the positions of the targets and the UAVs.

5.2 Effect of Increase in Number of Agents and Targets

A set of 100 simulations were carried out using PTCFA, OCFA and PSO solutions on a 1,000 × 1,000 m area. For the simulations using PTCFA and OCFA, the parameters for the UAVs are a velocity of 10 m/s, minimum turning radius of 50 m, a sensor range of 100 m, and a simulation time of 1,000 s. While the simulation parameters for the PSO coalition formation algorithm are: $c_1 = 0.5$, $c_2 = 0.5$, $\chi = 1$, $V_{\max} = 100$, and the number of iterations = 5,000. For each simulation, the target positions and resources, and the UAV positions and resources are randomly generated. The resources for each target were randomly generated between 0 and 3, while the UAV resources were randomly generated between 0 and (number_of_targets)/2. So, the minimum resources is equal to zero, while the UAV can carry a maximum of three different types of resources and each resource quantity can be equal to (number_of_targets)/2. We controlled the quantity of resources for the UAVs based on the number of available targets. For a given number of targets, we find the performance in terms of mission completion time by varying the number of agents to 5, 10, 15 and 20, while changing the number of targets to 5, 10, and 15. We also study the effect of percentage mission accomplished, time taken to form the coalition using PTCFA and OCFA (coalition time), and the CPU time required to carry out each simulation (simulation time).

The resources for the agents and the targets are generated randomly and the simulations are carried out in different ways for the two-stage and the PSO solutions. For the simulations using PTCFA and OCFA, the total time required to prosecute all the targets is recorded as the mission time. If some of the targets were not prosecuted due to lack of UAV resources then the percentage of the mission accomplished is recorded and calculated as:

$$\%\ \text{mission completed} \;=\; 100 - \frac{\text{Total number of target left}}{\text{Total number of targets}} \times 100.$$

The UAV team does not have *a priori* knowledge of whether the cumulative resources of the team are sufficient to prosecute all the targets or not. When the resources are not sufficient then UAVs have to search till the end of simulations, that is 1,000 s, otherwise the time taken to prosecute all the targets is recorded as mission

time. We also record the time to form the coalition and the CPU time needed to perform the simulation.

For the PSO based optimal solution, each simulation was carried out by running the PSO algorithm for a given number of iterations (5,000) and the mission time obtained from the solution and the CPU time taken for the solution are stored. When the cumulative UAV resources are less than the cumulative target resources then a solution using PSO cannot be obtained and the percentage mission accomplished is 0% and the mission time in that case is treated as 1,000 s.

The performance curves for the Monte-Carlo simulations are shown in Figs. 12, 13 and 14. Let us first consider a mission with five targets and varying number of UAVs as shown in Fig. 12a. From the figure, we can see that the optimal solution obtained from PSO outperforms the solution obtained using two-stage algorithms. However

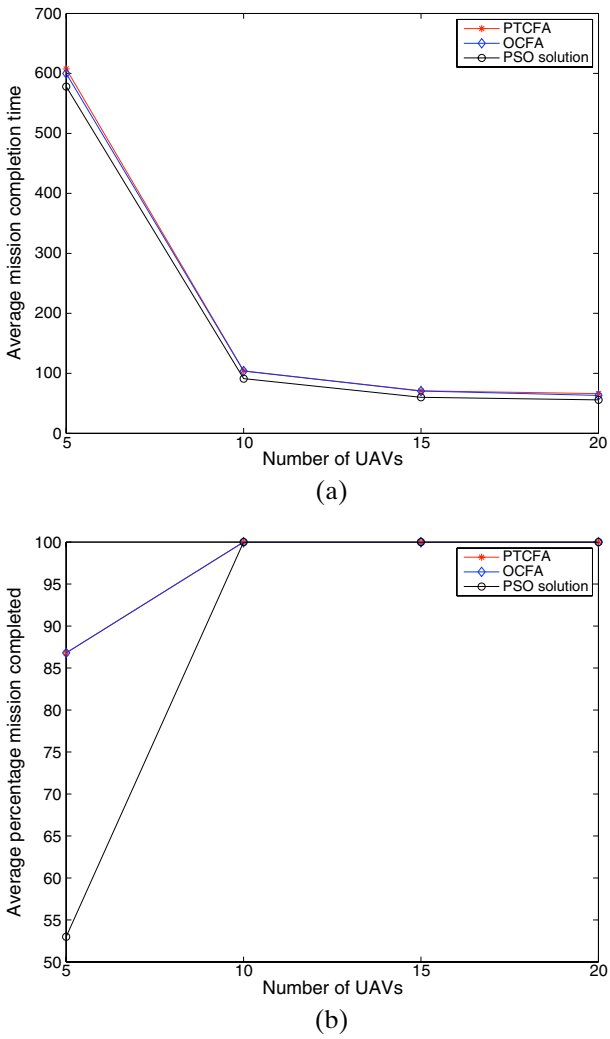


Fig. 12 **a** Comparison of mean mission completion time in seconds for a mission with five targets and increasing number of agents. **b** Mean percentage mission not accomplished due to lack of sufficient resources with increasing number of agents

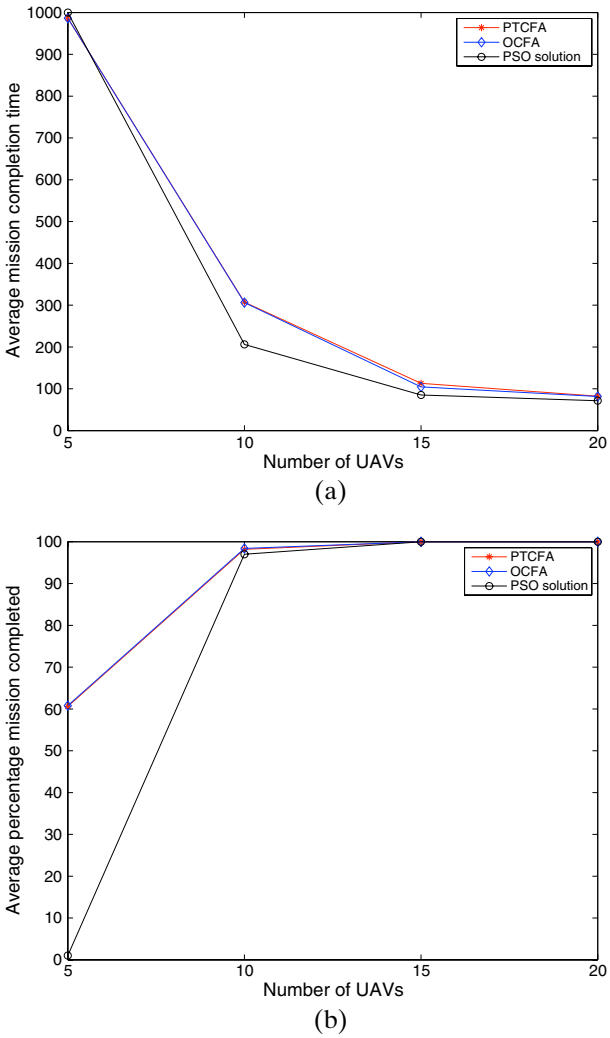
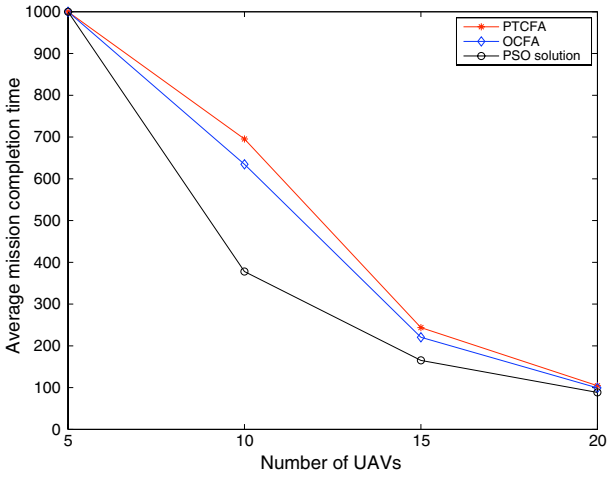


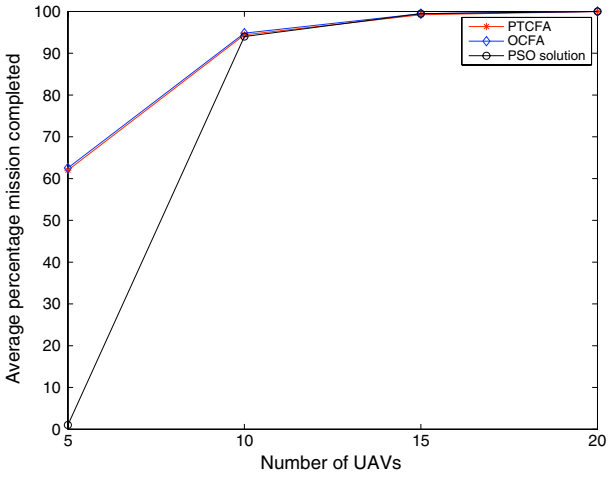
Fig. 13 **a** Comparison of mean mission completion time in seconds for a mission with ten targets and increasing number of agents. **b** Mean percentage mission not accomplished due to lack of sufficient resources with increasing number of agents

the deviation of the performance by PTCFA and OCFA algorithms from that of the optimal is marginal. The time to accomplish a mission has two components (i) the search time to detect the target and (ii) the time to prosecute the target. As the agents have small sensor range, they take more time to detect all the targets. Also, the target detection time depends on the kind of strategy used by the agents. Because of the lower sensor range and randomized search strategy, the mission completion time is higher for the two-stage algorithms. The mission time using these algorithms can be reduced by increasing the sensor range and also using better search scheme that can detect the targets early.

Figure 12b shows the percentage of the missions that were completed. From the figure we can see that the percentage mission completed when we use PSO solution is low for five UAVs compared to that obtained using the PTCFA and OCFA



(a)



(b)

Fig. 14 **a** Comparison of mean mission completion time in seconds for a mission with 15 targets and increasing number of agents. **b** Mean percentage mission not accomplished due to lack of sufficient resources with increasing number of agents

algorithms. This is because for some of the Monte Carlo runs, the UAVs did not have sufficient resources to prosecute all the targets. In those cases the problem does not have an optimal solution leading to 0% mission completion, and hence decrease in the percentage mission completed. But, for PTCFA or OCFA algorithms, the targets are prosecuted when detected and hence the percentage mission accomplished is always higher than that of PSO solution. When the number of UAVs are increased to 10, 15 and 20, all the targets were prosecuted by the UAVs, therefore the mean percentage mission completed is 100%.

The mission performance for ten targets is shown in Fig. 13a. For the mission with five UAVs, 99% of the simulations did not have sufficient UAV resources to prosecute all the targets. Hence, the average time for the mission is high. However, for the simulations with increasing number of UAVs, the mission time decreases.

The PSO solutions are better than the PTCFA and OCFA solutions. Although the PSO solution is better in terms of mission completion time, the percentage mission accomplished is lower than that achieved using PTCFA and OCFA algorithms. This phenomenon can also be seen for 15 targets in Fig. 14a, b.

From these simulations we can see that even though the PTCFA produces sub-optimal coalitions in polynomial time still its performance is comparable to that achieved using OCFA and PSO solution. As discussed previously, when a coalition leader receives proposals for possible coalition members then it uses PTCFA or OCFA to determine the coalition for the task. Figure 15 shows the mean computational time taken to form coalitions for various target and UAV distributions. From the figure we can see that the time taken for OCFA is higher than that using PTCFA for any given agent and target distribution. However, the time taken to solve these algorithms is less than one millisecond for both the algorithms which is encouraging for real-time applications.

With increase in the number of agents or targets, the simulation time increases when PSO algorithm is used. Figure 16 shows the average time taken (with standard deviation bars), to complete one simulation using PSO. For a given number of targets, the total number of computations increases with increasing number of agents and hence the time to determine the optimal solution also increases. However while using the two-stage algorithms it was found that the time to simulate was less than 30 s for all the cases that we studied.

5.3 Discussion

The simulation results show that the combinatorial optimization solution using PSO provides the lowest mission completion time. But, if there is an uncertainty in the number of resources allocated to UAV or in the information about the targets, then the solution obtained using combinatorial optimization is not valid. The advantages of forming coalitions in a decentralized manner is that the agents need not have any *a priori* information about other members in the team or the targets. The information is required only while forming coalitions. Thus, even if the resources of some of the

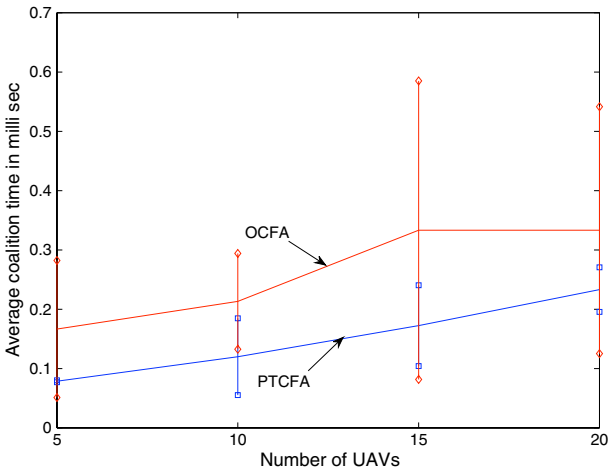


Fig. 15 Average computational time taken to form coalitions for given number of targets and varying number of agents

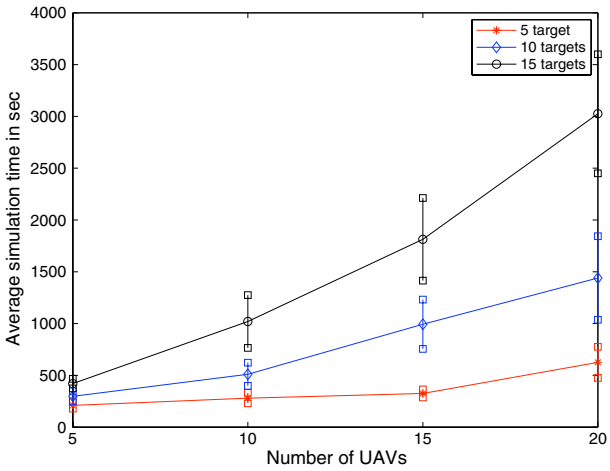


Fig. 16 Mean time taken to complete a simulation using PSO

UAVs are exhausted or cannot be used for some reason, the decentralized coalition formation algorithms still can handle these uncertainties.

5.3.1 Coalition Formation for Agents with Heterogenous Speed

In the simulations, we assumed that the UAVs have same velocity. However, the developed coalition formation algorithms can generate coalitions when the agents have different velocities without any change to the algorithm. This is because, the algorithm uses ETA of the agents which implicitly captures their velocity information. We present an example with five agents and three targets. The initial locations of the UAVs and the targets are shown in Fig. 17a and agents use the PTCFA algorithm to determine coalitions. The agent speeds are $v_1 = 8$ m/s, $v_2 = 10$ m/s, $v_3 = 13$ m/s, $v_4 = 15$ m/s, and $v_5 = 18$ m/s. The agent resources are $\mathcal{R}^{A_1} = (1, 1, 1)$, $\mathcal{R}^{A_2} = (2, 1, 1)$, $\mathcal{R}^{A_3} = (2, 2, 1)$, $\mathcal{R}^{A_4} = (2, 2, 1)$, $\mathcal{R}^{A_5} = (1, 1, 1)$ and the target resources are $\mathcal{R}^{T_1} = (1, 3, 1)$, $\mathcal{R}^{T_2} = (3, 3, 1)$, $\mathcal{R}^{T_3} = (3, 1, 3)$. The rest of the simulation parameters are same as described in Section 5.2.

At time $t = 0.4$ s, agent A_3 detects target T_1 and forms a coalition of agents A_4 and A_5. The trajectories followed by the agents A_4 and A_5 to prosecute T_1 is shown in Fig. 17b. As the mission progress, agent A_2 detects target T_3 and forms a

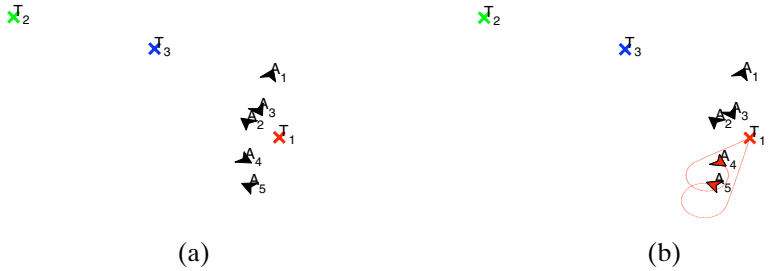
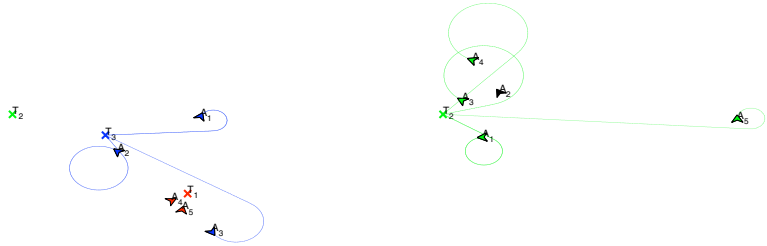


Fig. 17 **a** Initial agent and target locations. **b** The trajectories of agents A_4 and A_5 to target T_1



(a) (b)

Fig. 18 **a** Trajectories of agents A_1, A_2, A_3 to T_3. **b** Trajectories of agents A_1, A_3, A_4, and A_5 to T_2

coalition consisting of agents A_1, A_2 and A_3 at time $t = 23.1$ s. The trajectories of the agents are shown in Fig. 18a. The agents A_4 and A_5 prosecute T_1 at $t = 27.4$ s and subsequently they prosecute T_3 at time $t = 98.6$ s. The final target T_2 is detected by agent A_3 and forms a coalition with agents A_1, A_3, A_4, and A_5 at time $t = 123$ s. The target was prosecuted at time $t = 198.2$ s and the trajectories of the agents are shown in Fig. 18b. From the trajectories, we can see that even though the initial ETAs of the agents are different, the simultaneous strike mechanism ensures that the agents reach the target at the specified time which is the maximum of ETAs of all the agents in the coalition.

5.3.2 Minimum Member Coalition

The coalition formation algorithm presented here (OCFA) can be extended to a single objective of minimizing the total number of UAVs, and not taking the time to accomplish the mission into account. In this case, the first stage of the two-stage algorithm can be removed, and solving only the second stage of OFCA will yield the desired result.

5.3.3 Improving the Performance of PTCFA

The second stage of the polynomial time algorithm attempts to determine the minimum number of agents required for the coalition through a systematic pruning of the coalition formed during stage 1. However, this algorithm is naive and we can incorporate different mechanisms that can provide better solutions. A mechanism can be to sort the agents by their resource contribution and then screen them, in which case, the agents with low resources will be released from the coalition that might result in a coalition with lesser members. However, there is no guarantee that even such a mechanism would produce an optimal member coalition.

5.3.4 Other Applications

The coalition formation algorithm can be used for many other applications where a sub-team of UAVs are required to perform a task. These include cooperative tracking, patrolling sea shores, and cooperatively tracking and prosecuting moving targets in a battle-field. In these applications, the coalitions are formed to share sensor information and the resources may not deplete. The proposed coalition formation algorithms can be easily modified depending on the mission. For example,

consider a cooperative target tracking mission where the coalition leader detects a target and assigns a coalition of UAVs to track the target with varying sensors on board and prosecute it [22].

6 Conclusion

In this paper, we presented two decentralized two-stage coalition formation algorithms, the first one has polynomial time complexity and the second uses binary integer programming technique with low computational overhead, for a search and prosecute mission. The algorithms determine optimal set of agents that have sufficient resources to simultaneously prosecute a target. In order to determine the deviation of the solution provided by the proposed algorithms from an optimal centralized solution, we developed a PSO formulation to solve the centralized combinatorial optimization problem. Simulation results are presented to show that the mission performances of the two-stage algorithms are close to the PSO solution. The simulation time taken by the proposed algorithms, as well as the time to form coalitions, is less and this holds promise for real-time applications.

Acknowledgements This work was partially funded by the National Science Foundation under Information Technology Research Grant CCR-0313056 and by NASA under STTR contract number NNA04AA19C to Scientific Systems Inc, and Brigham Young University and by the Air Force Office of Scientific Research award No. FA9550-04-0209.

References

1. Gerkey, B., Mataric, M.J.: A formal framework for the study of task allocation in multi-robot systems. Int. J. Robot. Res. **23**(9), 939–954 (2004)
2. Nygard, K.E., Chandler, P.R., Pachter, M.: Dynamic network flow optimization models for air vehicle resource allocation. In: Proc. of the American Control Conference, Arlington, Texas, pp. 1853–1858
3. Schumacher, C., Chandler, P.: UAV task assignment with timing constraints via mixed-integer linear programming. In: AIAA Unmanned Unlimited Technical Conference, Workshop and Exhibit, Chicago, Illinois. AIAA-2004-6410 (2004)
4. Darrah, M., Niland, W., Stolarik, B.: UAV cooperative task assignments for a SEAD mission using genetic algorithms. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, Colorado. AIAA-2006-6456 (2006)
5. Alighanbari, M., How, J.: Robust decentralized task assignment for cooperative UAVs. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, Colorado, 21–24 Aug 2006
6. Sujit, P.B., Sinha, A., Ghose, D.: Multi-UAV task allocation using team theory. In: Proc. of the IEEE Conference on Decision and Control, and European Control Conference, Seville, Spain, pp. 1497–1502 (2005)
7. Sujit, P.B., Sinha, A., Ghose, D.: Multi-UAV task allocation using negotiation. In: Autonomous Agents and Multi-Agent Systems, Hakodate, Japan, (2006)
8. Shehory, O.M.: Methods for task allocation via agent coalition formation. Artif. Intell. **101**(12), 165200 (1998)
9. Sandholm, T., Larson, K., Andersson, M., Shehory, O., Tohme, F.: Coalition structure generation with worst case guarantees. Artif. Intell. **111**(12), 209238 (1999)
10. Shehory, O.M., Sycara, K., Jha, S.: Multi-agent coordination through coalition formation. In: Rao, A., Singh, M., Wooldridge, M. (eds.) Lecture Notes in Artificial Intelligence, no. 1365. Intelligent Agents IV, pp. 143–154. Springer, New York (1997)
11. Vig, L., Adams, J.A.: Multi-robot coalition formation. IEEE Trans Robot **22**(4), 637–649 (2006)

12. Vig, L., Adams, J.A.: Market-based multi-robot coalition formation. In: Gini, M., Voyles, R. (eds.) Proc. of the International Symposium on Distributed Autonomous Robotic Systems, pp. 227–236. Springer, Minneapolis (2006)
13. Vig, L., Adams, J.A.: A framework for multi-robot coalition formation. In: Proc. of the Indian International Conference on Artificial Intelligence. India (2005)
14. Parker, L.E., Tang, F.: Building multi-robot coalitions through automated task solution synthesis. In: Proc. of the IEEE Special Issue on Multi-robot Systems, vol. 94, no. 7, pp. 1289–1305 (2006)
15. Lin, J., Morse, A.S., Anderson, B.D.O.: The multi-agent rendezvous problem. In: Proc. of the IEEE Conference on Decision and Control, Maui, Hawaii, pp. 1508–1513 (2003)
16. Tiwari, A., Fung, J., Carson, J.M., Bhattacharya, R., Murray, R.M.: A framework for Lyapunov certificates for multi-vehicle rendezvous problems. In: Proc. of the American Control Conference, Boston, MA, pp. 5582–5587 (2004)
17. Lin, Z., Francis, B., Maggiore, M.: Necessary and sufficient graphical conditions for formation control of unicycles. IEEE Trans. Automat. Contr. **50**(1), 121–127 (2005)
18. McLain, T.W., Beard, R.W.: Coordination variables, coordination functions, and cooperative timing missions. AIAA J. Guid. Control Dyn. **28**(1), 150–161 (2005)
19. Furukawa, T., Bourgault, F., Whyte, H.F.D., Dissanayake, G.: Dynamic allocation and control of coordinated UAVs to engage multiple targets in a time-optimal manner. In: Proc. of the IEEE Conference on Robotics and Automation, Barcelona, Spain, pp. 2353–2358 (2005)
20. Notarstefano, G., Bullo, F.: Distributed consensus on enclosing shapes and minimum time rendezvous. In: Proc. of the IEEE Conference on Decision and Control, San Diego, California, pp. 4295–4300 (2006)
21. Dubins, L.E.: On curves of minimal length with a constraint on average curvature and prescribed initial and terminal positions and tangents. Am. J. Math. **79**, 497–516 (1957)
22. Kingston, D.B., Schumacher, C.J.: Time-dependent cooperative assignment. In: Proc. of the American Control Conference, Portland, Oregon, pp. 4084–4089 (2005)
23. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proc. of the Symposium on Micro Machine and Human Science, Piscataway, NJ, p. 3943 (1995)
24. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. of the IEEE International Conference on Neural Networks, Piscataway, NJ, p. 1942–1948 (1995)
25. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: Proc. of the IEEE Conference on Evolutionary Computation, Anchorage, Alaska, pp. 69–73 (1998)
26. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through particle swarm optimization. In: Natural Computing, vol. 1, pp. 235–306. Springer, New York (2002)
27. Laskari, E.C., Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization for integer programming. In: Proc. of the IEEE Congress on Evolutionary Computation, Honolulu, pp. 1582–1587 (2002)
28. Nemhauser, G.L., Wolsey, L.A.: Integer programming. In: Nemhauser, G.L., Rinnoy Kan, A.H.G., Todd, M.J. (eds.) Handbooks in Operations Research and Management Science. Vol. 1: Optimization. Elsevier, Amsterdam (1999)
29. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Englewood Cliffs (1981)
30. Sujit, P.B., George, J.M., Beard, R.: Multiple UAV coalition formation. In: Proc. of the American Control Conference, Seattle, Washington (2008)