Solving Task Allocation Problem in Multi Unmanned Aerial Vehicles Systems Using Swarm Intelligence

Janaína Schwarzrock[a], Iulisloi Zacarias[a], Ana L. C. Bazzan[a], Edison P. de Freitas[a], Ricardo Queiroz de Araujo Fernandes[b], Leonardo Henrique Moreira[b]

[a]*Institute of Informatics Federal University of Rio Grande do Sul, Brazil*
[b]*Software Development Center - Brazilian Army, Brazil*

Abstract

The envisaged usage of multiple Unmanned Aerial Vehicles (UAVs) to perform cooperative tasks is a promising concept for future autonomous military systems. An important aspect to make this usage a reality is the solution of the task allocation problem in these cooperative systems. This paper addresses the problem of tasks allocation among agents representing UAVs, considering that the tasks are created by a central entity, in which the decision of which task will be performed by each agent is not decided by this central entity, but by the agents themselves. This is a reasonable assumption, given the way strategic planning is carried up in military operations. To enable the UAVs have the ability to decide which tasks to perform, concepts from swarm intelligence and multi-agent system approach are used. Heuristic methods are sometimes used to solve this problem, but they have drawbacks. For example, many tasks end up not begin performed even if the UAVs have enough resources to execute them. To cope with this problem, this papers proposes three algorithm variations that complement each other to form a new method aiming to increase the amount of performed tasks, so that a better task allocation is achieved. Through experiments in a simulated environment, the proposed methods were evaluated, yielding enhanced results for the addressed problem compared to

Email addresses: `jschwarzrock@inf.ufrgs.br` (Janaína Schwarzrock), `izacarias@inf.ufrgs.br` (Iulisloi Zacarias), `bazzan@inf.ufrgs.br` (Ana L. C. Bazzan), `epfreitas@inf.ufrgs.br` (Edison P. de Freitas), `ricardo@cds.eb.mil.br` (Ricardo Queiroz de Araujo Fernandes), `moreira@cds.eb.mil.br` (Leonardo Henrique Moreira)

existing methods reported in the literature.

Keywords: Unmanned Aerial Vehicles, Task Allocation, Multi-agent Systems, Swarm Intelligence

1. Introduction

The use of unmanned aerial vehicles (UAVs) to perform the so called dull, dirty and dangerous (3D) missions is becoming very common. There are many research focusing this theme, such as [1, 2, 3]. A special case is the use of UAVs for military purpose [4]. New applications with multiple UAVs have been planned [5, 6, 7, 8], in which UAVs cooperatively patrol perimeters, monitor areas of interest or escort convoys. Areas of difficult access, borderline regions and critical infrastructure are examples of application scenarios that are easier monitored by groups of UAVs.

Several kinds of sensors can be used for monitoring purposes, varying according to the situation. Each UAV can be equipped with one or more of these sensors, but they have limited resources such as time or energy (batteries or fuel that limit their endurance). In order to deploy an application in which a fleet of UAVs is designed to monitor a given area, these aspects have to be taken into account. If a massive usage of this type of system is considered, a centralized approach to allocate surveillance tasks to the UAVs does not scale.

In military operations it is common to have a central command unit that coordinates and delegates missions to be performed by military teams acting on the field. However, most commonly, the teams that receive these missions have autonomy to internally decide which members will perform the different parts of the mission. Observing this organization structure, this work focuses on teams of UAVs that must autonomously and cooperatively complete a mission assigned by the central commnad entity. The team of UAVs is seen as a group that receives a given mission, which contains a set of tasks, and internally has to take care of the division of it among its members.

This problem can be handled as a task allocation among agents and modeled using the generalized assignment problem (GAP), in which the UAVs are the agents and the mission is associated with a set of tasks. The GAP is known to be NP-Complete [9]. Using a swarm intelligence-based approach is an appropriate alternative to deal with this problem in a decentralized way. Thus, each UAV can decide which tasks it will perform considering only local information, such as its location and resources status.

In the related literature, there is a heuristic method for task allocation based on swarm intelligence, called Swarm-GAP[10] (see Section 3), which allows agents to perform task allocation in an autonomous and decentralized way. In this method, there is no central command unit that has knowledge of the set of tasks. Rather, these tasks are perceived by the agents and they "communicate" (pass information about perceived tasks) to other agents through a token-based communication protocol.

Swarm-GAP presents efficient results where the agents themselves perceive the tasks that need to be performed in the environment in which they act, create the tokens, and send them to other agents. Swarm-GAP works best where several tokens are created, each containing few tasks. However, when a token contains many tasks, as is the case of tokens created by a central command unit, Swarm-GAP is not suitable because it cannot make an efficient use of agents' resources. The consequence is that many tasks are not selected by the agents, even if they have enough resources to execute them.

As mentioned above, the assumption of the existence of such central command unit is reasonable: in the case of military operations, the central entity's role that creates the tasks is of capital importance because this entity has a holistic view of the situation, which facilitates strategic planning. In general, when it comes to tasks that are delegated by a central entity, a token contains multiple tasks. Since the Swarm-GAP algorithm is not efficient for this situation, there is a need to provide a more suitable solution.

Therefore, the contribution of this paper is to propose a new method with three variations based on Swarm-GAP, which aim to: (i) allow the agents use their resources to perform as many tasks as possible; (ii) avoid the agents assigning their resources to tasks that are not very suitable for them; and (iii) allow an efficient workload balance among the agents, that is, to prevent some agents from being overloaded with tasks while others remain idle.

The remainder of this paper is organized as follows. Section 2 states the problem statement. In Section 3, the Swarm-GAP algorithm is briefly presented. The proposed solutions are described in Section 4. The description of the experimental setup is described in Section 5. The experiments and their results are presented and discussed in Section 6. Section 7 presents the related works. Section 8 then makes concluding remarks and future directions.

2. Problem formulation

In the problem addressed in this work, each task created by the central command unit (henceforth referred simply as central) refers to the activity of monitoring an area with the objective of detecting some type of target. Each type of target can be detected by one or more sensors that a UAV has. However, each kind of sensor has a different quality in detecting the targets, i.e., some sensors are more suitable for some types of targets than others.

Furthermore, each UAV can be equipped with one or more sensors. Thus, each UAV can perform more than one type of task. However, it is desirable that the tasks be performed by the UAVs that are more suitable for them, that is, those equipped with sensors that offer higher quality to detect the type of target required by the task. Each task needs be performed by only one UAV, but each UAV can perform any or many tasks, as long as it has quality and resource to execute them.

The central has information about the tasks that need to be performed, but does not necessary knows the current status and positioning of the agents in the environment. As mentioned, a mission contains many tasks. It may happen that the UAVs cannot complete the whole mission, depending on the quantity of able agents and the time that is needed to execute the set of tasks that composes a mission.

The goal is to find, in a decentralized way, an appropriate task allocation among UAVs so that the quantity and quality of the performed tasks is maximized and the time spent on it is minimized. This problem can be modeled as a Generalized Assignment Problem (GAP), that is known to be a NP-complete problem [9]. The problem formulation is given as follows.

Let $\mathcal{I} = \{i_1, ..., i_m\}$ be a set of m UAVs and $\mathcal{S} = \{s_1, ..., s_u\}$ be a set of u types of sensors. Each UAV $i \in \mathcal{I}$ is modeled by a set of attributes $D_i = \{l, Si, r\}$, where l is the coordinates $\langle x, y \rangle$ of the UAV's current location; $Si \subseteq \mathcal{S}$ is a set of types sensors with the UAV is equipped and r is the available resource.

A mission M is composed by a set $\mathcal{J} = \{j_1, ..., j_n\}$ of n tasks and have a deadline dl, which determines the maximum time that UAVs have to complete the mission. Each task $j \in \mathcal{J}$ is modeled by a set of attributes $E_j = \{l, t, c\}$, where l is the coordinates (x, y) of the task's location; $t \in \mathcal{A} = \{a_1, ..., a_v\}$ is the type of target that needs to be detected; and c the amount of resource necessary to survey all task's area.

The matrix $QM = (q_{sa})_{u \times v}$ with $q_{sa} \in [0, 1]$, is called quality matrix, where the $(s, a)^{th}$ entry q_{sa} corresponds to the sensor s' quality to detected the type of target a. The quality of the UAV i to perform a task j, is given by Eq. 1. When $Q(i, j) = 0$ the UAV i no have sensor capable of detecting t_j.

$$Q(i, j) = \max_{s \in Si_i} q_{st_j} \tag{1}$$

Each agent i has a specific capability k_{ij} to perform each task j. The capability in this problem is defined by Eq. 2, where J is the set of available tasks, $d(i, j)$ is the Euclidean distance between the UAV and the task, $Q(i, j)$ is the UAV's quality to perform the task and $\alpha \in [0, 1]$ is the weight given to the distance and quality factors.

$$k_{ij} = \frac{\max_{g \in J}\{d(i,g)\} - d(i,j)}{\max_{g \in J}\{d(i,g)\}} \times \alpha + \\ (1 - \frac{\max_{g \in J}\{Q(i,g)\} - Q(i,j)}{\max_{g \in J}\{Q(i,g)\}}) \times (1 - \alpha) \tag{2}$$

According to this equation, the closer to a task location the UAV is, and the greater its sensors' quality to detect the type of target required by that task, the greater its capability.

The matrix $X = (x_{ij})_{m \times n}$, called allocation matrix, represents the task allocation among UAVs. Where, x_{ij} is 1 if the task j is allocated to UAV i and 0 otherwise. The goal is find a allocation matrix X that maximize the total reward, which is given by agent's capabilities (Eq. 3),

$$X = \arg\max_{X'} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} k_{ij} \times x'_{ij} \tag{3}$$

subject to constrains:

$$\forall j \in \mathcal{J} \sum_{i \in \mathcal{I}} x_{ij} \leq 1 \tag{4}$$

$$\forall x \in X \mid x_{ij} = 1, Q(i,j) > 0 \tag{5}$$

$$\forall i \in \mathcal{I} \sum_{j \in \mathcal{J}} x_{ij} \times C_{ij} \leq r_i \tag{6}$$

The constrain in Eq. 4 ensures that each task is allocated to at most one UAV. Eq. 5 restricts the allocation of the tasks only to the agents that have the quality to execute them. Finally, the constrain in Eq. 6, requires that it must respect the UAV's resources limitations.

For simplicity, a single type of resource is being considered: the time. Then, the resources r_i for all UAV i is the same as the mission deadline dl_M. When a task j is performed by UAV i, it consumes C_{ij} units of r_i. $C_{ij} = d(i,j) + c_j$, where $d(i, j)$ is the distance traveled up by UAV i to the task j and c_j is the time to execute the task.

3. Background

The authors in [10] have proposed the Swarm-GAP, an algorithm for distributed task allocation based on theoretical models of division of labor in social insect colonies. As the work here proposed is based on the Swarm-GAP, this section briefly presents an overview of this algorithm.

Swarm-GAP allows that each agent chooses which tasks it will perform. This decision is based just on local information, thus leading to little communication. This method for tasks allocation is based on a probabilistic approach, by using the mathematical response threshold model formulated by Theraulaz et al. in [11]. The task allocation is modeled as a generalized assignment problem (GAP) and the goal is to maximize the total capability.

Swarm-GAP uses a communication model based on a token passing protocol, as can be observed in its pseudo code presented in Algorithm 1. When an agent receives a token (line 1), it decides which tasks it will perform (line 3 to 8). The agent likelihood to choose a task is determined by the tendency $\mathrm{T}_{\theta_{ij}}$ (Equation 7). This tendency is calculated based on the task's stimulus st_j and on a threshold θ_{ij}. The threshold is based on the agent's capability to perform a task (Equation 8). The higher the capability, the lower the threshold.

$$\mathrm{T}_{\theta_{ij}}(st_j) = \frac{st_j^2}{st_j^2 + \theta_{ij}^2} \tag{7}$$

$$\theta_{ij} = 1 - k_{ij} \tag{8}$$

The decision also depends on the available resources (line 6), i.e., the agent must have enough resource to perform the task. When the agent decides to carry out a task, this task is allocated to the agent (line 7) and the agent's resources are reduced accordingly (line 8). After that, the agent is marked as visited (9) and if there are still unallocated tasks, the agent sends the token to an agent which has not yet received that specific token (line 10 to 11).

Algorithm 1: Pseudo code of the Swarm-GAP

```
1: Receive Token
2: Compute available resources $r_i$
3: for all available tasks do
4:     Compute capability $k_{ij}$
5:     Compute tendency $T_{\theta_{ij}(st)}$
6:     if roulette() $< T_{\theta_{ij}(st)}$ and $r_i \geq c_j$ then
7:         Allocate task $j$ to agent $i$
8:         Decrease resource $r_i$
9: Mark agent as visited in the token
10: if there are still available tasks then
11:     Send the token to a not yet visited agent
```

Swarm-GAP also allows that an agent creates a token when it perceives a task in the environment that needs to be done. This feature is not further discussed here because, in the context of this work, the token with the tasks is only created by a central entity. For more details about Swarm-GAP, see [10] or [12].

4. Proposed solution

The proposal based on Swarm-GAP presented in this work is divided in three variations. This section describes each of them as well as the motivation that explain their proposals.

4.1. Allocation Loop (AL)

When Swarm-GAP is used to solve the task allocation problem presented in Section 2, the resources of the UAVs were not taken fully into account: many tasks were not performed and resources were underused. To take advantage of the available resources of the UAVs and maximize the amount of tasks that are performed, an Allocation Loop (AL) algorithm was proposed.

Algorithm 2 provides the pseudo code of this method. In AL, instead of dropping the token after all UAVs have received it, the list of visited agents in the token is cleaned and a new token-sending round is started. Thus, each UAV may receive the token more than once and the unallocated tasks will have a new chance of being allocated. However, a simple cleaning scheme of this list may not be effective. When a token still has unallocated tasks, but the UAVs already allocated all their resources, the token is unnecessarily kept in circulation, thus causing an unnecessary exchange of messages among the UAVs.

To prevent the token remain circulating forever, after cleaning the list of visited agents, the agents that still have capability or available resources for executing tasks, are inserted in this list again. Then, only agents with some probability of allocating tasks will may receive the token in next round. For this, when a UAV receives a token, after selecting the tasks and mark itself as visited (lines 1 to 3), it informs the token if it has or does not have availability to carry out any of the remaining tasks (line 5). Then, the token stores a list of agents that should no longer receive the token in the next rounds, called list of unavailable agents, because they can no longer select any other task contained in the token.

4.1.1. Sorting and Allocation Loop (SAL)

In previous method, since the choice of tasks is done in the order that the tasks are in the token, often the UAVs end up selecting first the tasks that are not those more suitable for them. Then, it may cause lack of resource for others tasks more appropriate for them. To deal with this problem, a sorting mechanism that complement the AL is proposed. This algorithm is called Sorting and Allocation Loop (SAL).

The SAL's pseudo code is presented in Algorithm 3. In SAL, the tasks are sorted by tendency in decreasing order (line 3 to 6). This facilitates the selection of a task by the UAV i.e. it makes easier to the UAVs select the more

Algorithm 2: Pseudo code - Allocation loop (AL)

```
1: Receive Token
2: Select the tasks that it will perform (lines 2 to 8 of the Algorithm 1)
3: Mark agent as visited in the token
4: if there are still available tasks then
5:     Inform token if it has availability to perform any one of these tasks
6:     if all agents already receive the token then
7:         Clean the list of visited agents
8:         Fill list of visited agents with the unavailable agents
9:     Send the token to a not yet visited agent
```

appropriate tasks to them. This sort gives priority to the tasks with the greatest tendency, avoiding the selection of tasks with lower tendency.

Algorithm 3: Pseudo code - SAL

```
1: Receive Token
2: Compute available resources $r_i$
3: for all available tasks do
4:     Compute capability $k_{ij}$
5:     Compute tendency $T_{\theta_{ij}}$
6: Sort tasks by descending tendency
7: for all available tasks sorted by tendency do
8:     The same as lines 6 to 8 of the Algorithm 1
9: The same as lines 3 to 9 of the Algorithm 2
```

4.1.2. Limit and Allocation Loop (LAL)

By the way it was conceived, the SAL algorithm causes an increase in the amount of performed tasks. Nevertheless, it was observed in the experiments that, in some runs, there were UAVs that still remained idle, i.e, UAVs that performed no task. When the UAVs have enough resource, the first one visited by the token allocates most of the tasks. Therefore, it may result in idleness for other UAVs.

For this reason, the Limit and Allocation Loop (LAL) method adds to SAL a selection limit per UAV in each round. This limit means that a UAV cannot allocate more than one task each time it receives the token. Thus, all UAVs have the chance to choose a task within a round and the workload is in result more balanced.

5. Experimental setup

In order to evaluate the proposed solutions to the task allocation problem among UAVs, it was implemented a simulation in NetLogo[13] (version 5.3.1). The tasks and UAVs are represented, each one, by a specific symbol. Figure 1 illustrates the simulation graphical interface, where the symbol for the tasks is represented by a "X" shape and the symbol for a UAV has the shape of an airplane.

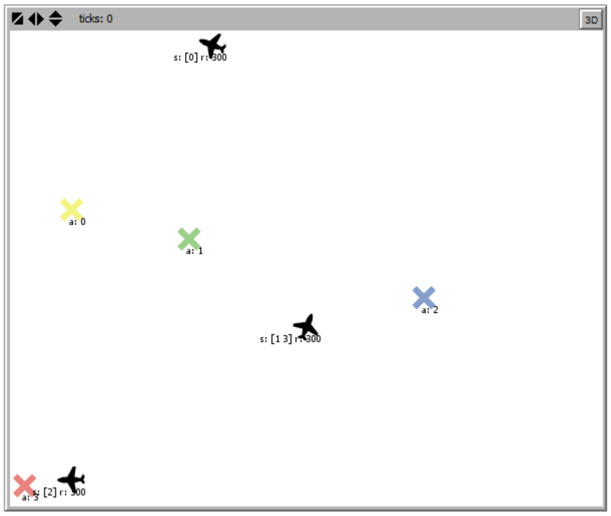

Figure 1: Simulation in the NetLogo

According to the formulated problem (Section 2), each task has its location (x,y), an associated type of target and the amount of resource required for its execution. The difference in the colors of the tasks represents the different types of targets. Each UAV, in addition to its location (x,y) and its sensor list, has a "to-do" list of tasks. This list starts empty, and when the UAV selects a task, it adds the task to this list. The "to-do" list is a FIFO: the first selected task is the first one to be performed.

The simulation was implemented so that at each tick, all UAVs move one pixel. The ticks are used as makespan (total time that elapses from the beginning to the end of the execution). It will never exceed the mission deadline. The execution ends when the UAVs complete all tasks they have selected to do. It is noteworthy that the makespan may be easily translated into traveled distance by each UAV since all UAVs move one pixel per tick.

Only one token is released to the UAVs. The token contains the mission tasks and a list of visited agents. This list starts empty and each UAV that receive the token add its ID in it. At each tick, the token is sent to a random UAV that has not yet received the token in the current round, ie an UAV that is not on the visited agents list. The token also has a list of unavailable agents. This list is used by algorithms AL, SAL and LAL (see Section 4.1). When an UAV receives the token, it runs the algorithm to choose the tasks that it will carry out.

Several experimental scenarios were created in order to analyze different situations. The scenarios vary in amount of mission tasks, number of agents and area size (pixels) in which they are situated. As described as follows.

i) 3 UAVs; 4 tasks; 300 ticks as deadline; 100 x 80 px area size.

ii) 3 UAVs; 8 tasks; 300 ticks as deadline; 100 x 80 px area size.

iii) 3 UAVs; 16 tasks; 300 ticks as deadline; 100 x 80 px area size.

iv) 3 UAVs; 32 tasks; 300 ticks as deadline; 100 x 80 px area size.

v) 6 UAVs; 64 tasks; 300 ticks as deadline; 200 x 160 px area size.

vi) 9 UAVs; 96 tasks; 300 ticks as deadline; 300 x 240 px area size.

Each set of UAVs – with 3, 6, and 9 UAVs – were randomly created once. For example, the set of 3 UAVs were created once and used in scenarios i, ii, iii, iv. The UAVs location (x,y) were defined by random values, respecting the limit of the scenario's area. The number of sensors that each UAV was equipped as well as the types of sensors were also assigned with random values. Each UAV was equipped with one or two sensors of types s_0, s_1, s_2 or s_3.

The same was applied for the tasks. The tasks location (x,y) and the type of target were defined by random values. Each task was assigned a single type of target, which may be of the types a_0, a_1, a_2 or a_3. Only the amount of resources needed to execute a task (c_j) was fixed at ten time units (ticks) for all.

Table 1 presents the sensors' quality to detect the each type of target. This values were used in all scenarios. According to this configuration, the target a_0, for example, can be detected by the sensors s_0 and s_2, but with different qualities.

Table 1: Sensors quality to detect the types of target

Sensor / target	a_0	a_1	a_2	a_3
s_0	1.0	0	0.3	0.5
s_1	0	0	1.0	0
s_2	0.2	0	0	1.0
s_3	0	1.0	0	0.3

The scenarios were run 30 times for each algorithm. In all experiments, the UAV's capability was computed using Equation 2 with $\alpha = 0.6$, giving greater importance to the distance factor. Different stimulus values were also tested, but only results with the value 0.6 are shown, as this value presented better results in most cases, and also agrees with what is known for Swarm-GAP.

The experiments were conducted in a PC with 1.70GHz, 4GB of RAM and SO Windows 8.1 Pro 64 bits. The results obtained using the proposed solutions as well as Swarm-GAP are presented and discussed in next section.

6. Experimental results and analysis

In order to evaluate the proposed methods, the following measures were considered: (a) Total reward – the main objective is to maximize it; (b) Makespan, which is the total time that elapses from the beginning to the end of the execution; (c) Quantity of tasks that were completed; (d) Quality of completed tasks; (e) Number of exchanged messages – related to the passing of the token; and (f) Algorithm's runtime.

These measures are presented to the methods AL, SAL and LAL and compared to those obtained using Swarm-GAP. The complete results with all measures (mean and standard deviation) of the scenarios i to vi are presented in the Tables 2 and 3. However, only a specific set of these scenarios is used to highlight the results for each measure. After presentation, the results are discussed. At the end of this section, an analysis of the scalability of the algorithms is also done.

In the following, the scenarios i, ii, iii and iv, which are composed of 4, 8, 16 and 32 tasks, respectively, are used to demonstrate the results of the total reward, makespan, quantity and quality of completed tasks and number of exchanged messages. This four scenarios were picked out because have the same number of UAVs and the amount of tasks is varied, but with the same mission's deadline. It allows to evaluate situations in which the UAVs have enough or more than enough time and the environment has few tasks (scenarios i and ii), and the UAVs have little time and there are many tasks to be performed (scenarios iii and iv). In the former, the UAVs can perform all tasks, but they must do it in the shortest time. On the other hand, in the second situation the UAVs do not have enough time to perform all tasks. Then, they must perform as many tasks as possible.

6.1. Total reward

The total reward of each method for different scenarios is shown in Figure 2. In the scenarios with 4 and 8 tasks, AL algorithm outperformed the Swarm-

Table 2: Results of algorithms in scenarios with 3 UAVs and different number of tasks

	Swarm-GAP	AL	SAL	LAL
	Mean (St.Dev.)	Mean (St.Dev.)	Mean (St.Dev.)	Mean (St.Dev.)
3 UAVs and 4 tasks in area of 100x80 pixels with deadline of 300 ticks				
Total reward	1.6971 (±0.4705)	2.1016 (±0.3251)	2.0947 (±0.3579)	2.2377 (±0.3217)
Comp. tasks (norm)	0.7583 (±0.1796)	1.0000 (±0.0000)	1.0000 (±0.0000)	1.0000 (±0.0000)
Elapsed time (norm)	0.3559 (±0.1874)	0.431 (±0.1388)	0.4139 (±0.1486)	0.3118 (±0.0759)
Quality (norm)	0.8131 (±0.1784)	0.8242 (±0.1284)	0.7942 (±0.1488)	0.9167 (±0.1428)
Idle UAVs	0.9333 (±0.5833)	0.5333 (±0.5074)	0.5667 (± 0.504)	0.1667 (± 0.379)
Sending token	2.8667 (±0.3457)	4.2000 (±1.7889)	4.0333 (±1.7317)	6.9667 (±2.2816)
Cost = (t + me) / ct	36.1433	33.3750	32.0500	25.1250
Total Runtime (sec)	2.1047 (±1.5154)	2.6855 (±2.1459)	3.4192 (±3.5977)	3.9216 (±2.9873)
3 UAVs and 8 tasks in area of 100x80 pixels with deadline of 300 ticks				
Total reward	2.9436 (±0.4874)	3.651 (±0.2064)	3.6006 (±0.3246)	4.8462 (±1.3581)
Comp. tasks (norm)	0.7708 (±0.1316)	0.9958 (±0.0228)	0.9833 (±0.0432)	1.0000 (±0.0000)
Elapsed time (norm)	0.6881 (±0.1722)	0.7731 (±0.1103)	0.7382 (±0.1264)	0.6786 (±0.0676)
Quality (norm)	0.7502 (±0.1599)	0.7699 (±0.0892)	0.7273 (±0.1003)	0.8125 (±0.1247)
Idle UAVs	0.3000 (±0.4661)	0.1667 (±0.3790)	0.2000 (±0.4068)	0.0000 (±0.0000)
Sending token	2.9667 (±0.1826)	6.8000 (±2.6182)	6.0000 (±2.3781)	11.1667 (±2.3057)
Cost = (t + me) / ct	33.9566 0	29.9664	28.9151	26.8417
Total Runtime (sec)	2.0803 (±0.6082)	2.7879 (±0.8128)	3.4072 (±1.3381)	4.8462 (±1.3581)
3 UAVs and 16 tasks in area of 100x80 pixels with deadline of 300 ticks				
Total reward	6.6424 (±1.0648)	7.6047 (±0.8502)	8.7607 (±0.6316)	10.5753 (±0.4666)
Comp. tasks (norm)	0.7229 (±0.0815)	0.8604 (±0.0585)	0.9313 (±0.0474)	0.9896 (±0.0288)
Elapsed time (norm)	0.9179 (±0.0495)	0.9462 (±0.0354)	0.9210 (±0.0496)	0.9027 (±0.0599)
Quality (norm)	0.8110 (±0.1040)	0.7431 (±0.0990)	0.7849 (±0.0591)	0.9239 (±0.0588)
Idle UAVs	0.0333 (±0.1826)	0.0000 (±0.0000)	0.0000 (±0.0000)	0.0000 (±0.0000)
Sending token	3.0000 (±0.0000)	5.7333 (±2.5316)	6.3667 (±1.4259)	18.7667 (±1.9420)
Cost = (t + me) / ct	24.0662	21.0363	18.9709	18.2885
Total Runtime (sec)	2.6142 (±0.8415)	2.8474 (±1.0097)	3.2888 (±0.9664)	6.7701 (±2.5940)
3 UAVs and 32 tasks in area of 100x80 pixels with deadline of 300 ticks				
Total reward	9.1017 (±1.4553)	9.0073 (±1.2457)	13.7469 (±0.8765)	19.9057 (±0.4189)
Comp. tasks (norm)	0.4781 (±0.0533)	0.4844 (±0.0498)	0.6094 (±0.0391)	0.7604 (±0.0222)
Elapsed time (norm)	0.9624 (±0.0177)	0.9652 (±0.0185)	0.9554 (±0.0168)	0.9422 (±0.0186)
Quality (norm)	0.7416 (±0.0723)	0.7275 (±0.0910)	0.8741 (±0.0512)	0.9315 (±0.0292)
Idle UAVs	0.0000 (±0.0000)	0.0000 (±0.0000)	0.0000 (±0.0000)	0.0000 (±0.0000)
Sending token	3.0000 (±0.0000)	3.5333 (±0.8996)	3.8333 (±0.9129)	24.8000 (±1.4716)
Cost = (t + me) / ct	19.0675	18.9097	14.8957	12.6356
Total Runtime (sec)	2.9778 (±0.7843)	3.2434 (±1.1824)	4.0268 (±1.0639)	10.9934 (±2.1070)

GAP by about 24% in both. SAL demonstrated similar results to AL, in this two scenarios (with 4 and 8 tasks).

As the amount of tasks increases, SAL allows a greater total reward than AL. SAL outperformed Swarm-GAP by 31.89% and 51.03% in the scenarios with 16 and 32 tasks, respectively. While AL overcomes the Swarm-gap by 14.48% in scenario with 16 tasks. AL and Swarm-GAP presented the same results to scenario with 32 tasks.

The highest total reward was obtained by the LAL algorithm. This variation outperformed Swarm-GAP by 31.85%, 64.63%, 59.2% and 118.7% in the

Table 3: Results of algorithms in scenarios with different number of UAVs and tasks

	Swarm-GAP	AL	SAL	LAL
	Mean (St.Dev.)	Mean (St.Dev.)	Mean (St.Dev.)	Mean (St.Dev.)
6 UAVs and 64 tasks in area of 200x160 pixels with deadline of 300 ticks				
Total reward	12.1152 (±1.9136)	12.9234 (±1.8407)	28.2929 (±1.0694)	38.7922 (±1.2800)
Comp. tasks (norm)	0.3135 (±0.0323)	0.3302 (±0.0317)	0.5271 (±0.0201)	0.6813 (±0.0208)
Elapsed time (norm)	0.9850 (±0.0086)	0.9819 (±0.0100)	0.9737 (±0.0101)	0.9643 (±0.0116)
Quality (norm)	0.7358 (±0.0790)	0.7447 (±0.0689)	0.9582 (±0.0319)	0.9667 (±0.0165)
Idle UAVs	0.0000 (±0.0000)	0.0000 (±0.0000)	0.0000 (±0.0000)	0.0000 (±0.0000)
Sending token	6.0000 (±0.0000)	6.5333 (±0.9371)	6.6333 (±1.0334)	44.0000 (±1.5086)
Cost = (t + me) / ct	15.0249	14.2477	8.8557	7.6445
Total Runtime (sec)	7.2348 (±3.4657)	10.7935 (±5.7570)	9.7962 (±4.7782)	31.9513 (±8.3246)
9 UAVs and 96 tasks in area of 300x240 pixels with deadline of 300 ticks				
Total reward	15.582 (±2.0050)	16.724 (±2.1908)	37.9608 (±1.1119)	44.733 (±1.5961)
Comp. tasks (norm)	0.2611 (±0.0217)	0.2774 (±0.0276)	0.4674 (±0.0170)	0.5226 (±0.0162)
Elapsed time (norm)	0.9886 (±0.0082)	0.9894 (±0.0064)	0.9763 (±0.0092)	0.9693 (±0.0124)
Quality (norm)	0.7899 (±0.0599)	0.7865 (±0.0641)	0.9680 (±0.0202)	0.9752 (±0.0159)
Idle UAVs	0.0000 (±0.0000)	0.0000 (±0.0000)	0.0000 (±0.0000)	0.0000 (±0.0000)
Sending token	9.0000 (±0.0000)	9.8667 (±1.1958)	9.7000 (±1.2360)	51.500 (±1.4797)
Cost = (t + me) / ct	12.1901	11.5157	6.7444	6.8233
Total Runtime (sec)	9.0990 (±4.6221)	13.1253 (±7.6605)	12.2062 (±2.7515)	50.5544 (±31.2161)

scenarios with 4, 8, 16 and 32 tasks, respectively. This pattern of increase in total reward can also be observed in the scenario with 64 and 96 tasks (See table 3). Where the LAL presented results 3.2× and 2.87× better than Swarm-GAP.

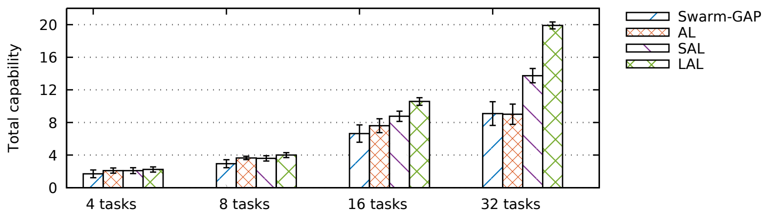

Figure 2: Total reward of each method for different scenarios with stimulus 0.6

6.2. Makespan and quantity of completed tasks

The quantity of completed tasks for each algorithm in each scenario is presented in Figure 3. It was normalized by the number of tasks of each scenario. Figure 4 shows the elapsed time (in ticks), normalized by the mission's deadline.

In the scenarios with 4 and 8 tasks, in which there is enough time (large deadline) for the UAVs carry out all tasks (the whole mission), the Swarm-

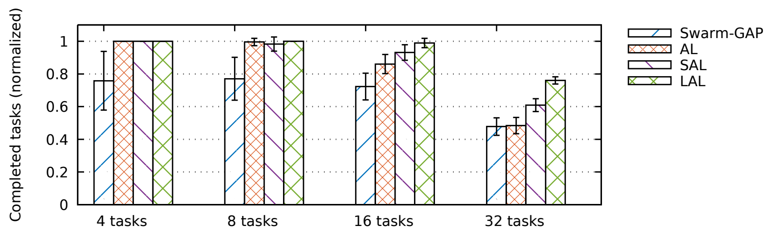

Figure 3: Completed tasks of each method for different scenarios with stimulus 0.6

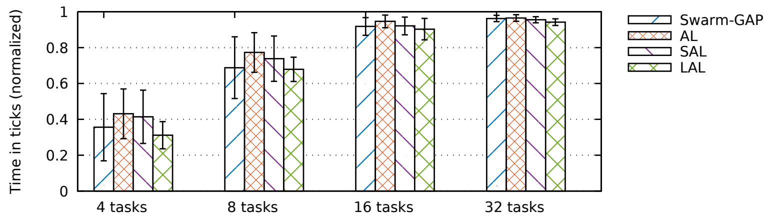

Figure 4: Makespan of each method for different scenarios with stimulus 0.6

GAP algorithm caused UAVs not to use all available resource. Moreover, more than 22% of the tasks were not performed in these two scenarios. The slack time (65% and 32%, respectively, in scenarios with 4 and 8 tasks) could have been used to perform other tasks. On the other hand, the proposition of using the Allocation Loop (AL) method has allowed 100% and 99% of the tasks to be carried out. This was possible because while an UAV still has resource it may select more tasks.

When the algorithms with Sorting and Allocation Loop (SAL) and Limit and Allocation Loop (LAL) were used, the UAVs have also performed all tasks in these scenarios with 4 and 8 tasks. However, they took less time than the AL. The LAL, for instance, improves the elapsed time by 27% and 12% compared to AL in the scenarios with 4 and 8 tasks, respectively.

In the scenarios with 16 and 32 tasks – i.e., there are more tasks to be performed –, the use of the SAL algorithm causes more tasks to be performed, in comparison to Swarm-GAP and AL, without increasing the elapsed time. For instance, in the scenario with 16 tasks, the use of SAL increased the amount of completed tasks by 28.8% compared to Swarm-GAP. The LAL method improves this performance even further. In the scenario with 16 tasks, sometimes the UAVs have performed all tasks when using the LAL algorithm. Furthermore, in the scenario with 32 tasks, LAL increased the completed tasks by 59%, whereas still reducing the elapsed time by 2.09% compared to Swarm-GAP.

6.3. Quality of completed tasks

The quality of the completed tasks in each scenario is shown in Figure 5. Quality was normalized to the amount of completed tasks. For this measure, the LAL algorithm also shown the best results.

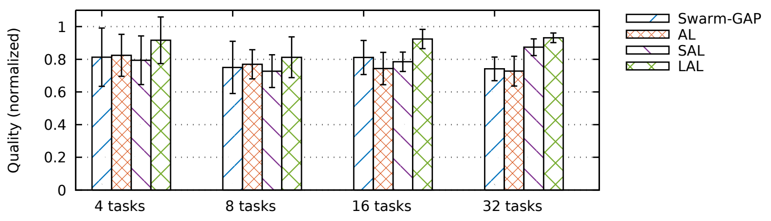

Figure 5: Quality of each method for different scenarios with stimulus 0.6

LAL have increased by 12.74% and 8.3% the completed tasks quality in the scenario with 4 and 8 tasks, respectively, compared to the Swarm-GAP algorithm. In these two scenarios, Swarm-GAP, AL and SAL presented similar results. Swarm-GAP outperformed SAL, but only with about 3%.

In the scenarios with 16 and 32 tasks, LAL presents an improvement of 13.92% and 25.6% compared to the Swarm-GAP, respectively. In the scenarios with 16, LAL still has a slightly lower result than Swarm-GAP (-3.2%). However, when there are more tasks, such as in the scenario with 32 tasks, LAL increased the quality by 17.86% compared to the Swarm-GAP. This pattern of quality improvement can also be observed in the other experiments with 64 and 96 tasks (see Table 3).

6.4. Number of exchanged messages

Figure 6 shows the number of exchanged messages for each algorithm in each scenario. It is possible to notice that this number is nearly constant in the Swarm-GAP, AL and SAL, while in the LAL method the number of exchanged messages increases linearly with the amount of completed tasks. The behavior of the three first algorithms is due to the fact that in these algorithms the number of sent tokens relates more to the amount of agents than to the number of performed tasks.

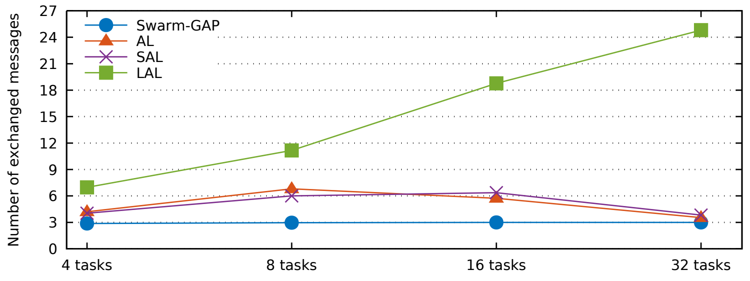

Figure 6: Total number of exchanged messages of each method for different scenarios with stimulus 0.6

With Swarm-GAP each UAV receive the token at most one time, so the token is sent 3 time if there are 3 UAVs. For AL and SAL, each UAV can receive the token more than once. On average each UAV receives approximately one to two times the token. Due to LAL's limitation of choice at most one tasks at time the UAV receive the token, the number of exchanged messages is about the same number of performed tasks.

Despite the increase in the number of exchanged messages in the LAL algorithm, the quantity of completed tasks improved and the elapsed time decreased. Hence, the LAL algorithm enables UAVs to spend less time in the execution of each task, i.e., the cost to carry out the tasks decreases.

In order to analyze if this increase in the number of exchanged messages is rewarded by the quantity of completed tasks, it is presented in the Figure 7 the average cost to perform each task, if it is was also considered the number of exchanged messages. Then, the cost was computed by $avgcost = (t + me)/ct$, where (t) is the elapsed time, (me) is number of exchanged messages and (ct) is the amount of completed task.

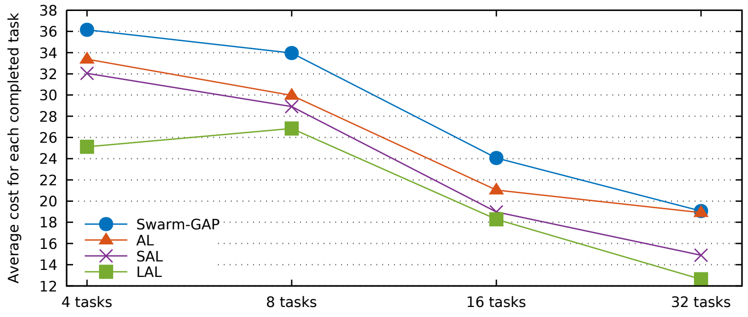

Figure 7: Average cost of the completed tasks (considering number of exchanged messages) of each method for different scenarios with stimulus 0.6

Even assuming that the amount of messages exchanged is also considered in the cost of executing the task, the LAL algorithm still reduces it compared to Swarm-GAP, by 30.48%, 20.95%, 24% and 33.73%, respectively for the scenarios with 4, 8, 16 and 32 tasks.

6.5. Algorithm's Runtime

To illustrate the results of the algorithms' runtime, the scenarios iv, v and vi, which are composed of 32, 64, and 96 tasks, respectively, were used. The amount of UAVs in these scenarios is varied, and the number of tasks is greater. Runtime was measured in seconds, using the NetLogo profiler extension.

The full results of these scenarios can be seen in the Tables 2 and 3. Here, the runtime analysis of each algorithm in different scenarios is highlighted in the Table 4. This table shows the mean total runtime in seconds, the number of times the algorithm was executed and the mean runtime of each execution.

The total runtime for AL and SAL are almost the same in all scenarios. This suggests that the sorting the tasks in SAL algorithm has not a great impact in the runtime. The AL and SAL results are close to Swarm-GAP in the scenario

Table 4: Runtime analysis of each method (in seconds) in different scenarios

Scenario		Swarm-GAP	AL	SAL	LAL
32 tasks	Total runtime (mean) in sec.	2.9778	3.2434	4.0268	10.9934
3 UAVs	Execution times (mean)	3.0000	3.5333	3.8333	24.8000
	Seconds per run	0.9926	0.9179	1.0504	0.4432
64 tasks	Total runtime (mean) in sec.	7.2348	10.7935	9.7962	31.9513
6 UAVs	Execution times (mean)	6.0000	6.5333	6.6333	44.0000
	Seconds per run	1.2058	1.6520	1.4768	0.7261
96 tasks	Total runtime (mean) in sec.	9.0990	13.1253	12.2062	50.5544
9 UAVs	Execution times (mean)	9.0000	9.8667	9.7000	51.5000
	Seconds per run	1.0110	1.3302	1.2583	0.9816

with 32 tasks. However, in the scenarios with 64 and 96 tasks, the availability check, that avoids the token remain in the network forever, impacts the runtime due to the greater number of tasks. For instance, the AL total runtime increases by 44.24% compared to Swarm-GAP in the scenario with 96 tasks.

On the other hand, the LAL increases the total runtime in all scenarios. Due to the selection limit, the LAL algorithm is executed more times causing the total runtime to increase. For example, in the scenario with 96 tasks, the LAL algorithm was executed on average 51.5 times – that is the same number of exchanged messages. Although the LAL presented a high total runtime, each UAV took 0.9816 seconds, on average, to execute the algorithm each time it received the token in this scenario with 96 tasks.

6.6. Discussion of results

The AL algorithm allows the UAVs perform more task than Swarm-GAP by enabling the UVAs select tasks while they still have resource to do the tasks. Consequently, the elapsed time increase, since more task are performed. This is a positive feature of our approach. Furthermore, the number of exchanged messages among UAVs, when used the AL, does not increase significantly. The AL algorithm also ensures that the token does not remain in the network forever (generating unnecessary communication) by informing the token when the UAV can not perform any of the tasks that are still available in the token. However, this availability check increases the runtime.

When the tasks are sorted by tendency (SAL algorithm) it is more likely that each UAV uses its time to perform the most suitable tasks. Conversely, when using Swarm-GAP and AL, an UAV may end up using its time to perform other, less appropriate tasks. Thus, more tasks are performed than AL, without increasing the elapsed time and with higher quality. This proves an increase in the total reward.

In addition to allowing more tasks to be performed and giving preference to using resources in more appropriate tasks, the LAL algorithm provides a better workload balancing by selection limit. Thus, it improves performance further. More tasks are performed in less time and the completed task quality increased even more. As each UAV can choose just one task at a time, it prevents some UAVs from selecting many tasks, becoming overloaded, while others UAVs remain idle.

By construction, LAL introduces more exchanged messages among UAVs. Nevertheless, it is counter-balanced by the amount of performed task in less time. The LAL drawback is its runtime. It improves the allocation and reduces the tasks execution cost, but increases the runtime, since it runs more times than other algorithms due to the selection limit.

6.7. Scalability analysis

In order to test the scalability of the proposed solutions, two new experimental scenarios were created. They have a very large number of agents and tasks (100 UAVs and 500 tasks), as described below. These scenarios differ from each other by the mission deadline.

vii) 100 UAVs; 500 tasks; 300 ticks as deadline; 750 x 750 px area size.

viii) 100 UAVs; 500 tasks; 1000 ticks as deadline; 750 x 750 px area size.

It is worth mentioning that the quantities used in this scenarios extrapolate those that would be applied to the addressed problem in this work in a real situation. However, the intention here was to stress the system to evaluate the reaction. The results are shown in Table 5.

Table 5: Results of algorithms in scenarios with 100 UAVs and 500 tasks

	Swarm-GAP	AL	SAL	LAL
	Mean (St.Dev.)	Mean (St.Dev.)	Mean (St.Dev.)	Mean (St.Dev.)
100 UAVs and 500 tasks in area of 750x750 pixels with deadline of 300 ticks				
Total reward	162.0141 (±6.5974)	162.9333 (±7.0324)	339.2015 (±7.5146)	240.956 (±3.6869)
Comp. tasks (norm)	0.4063 (±0.0141)	0.4090 (±0.0161)	0.7269 (±0.0159)	0.5022 (±0.0075)
Elapsed time (norm)	0.9979 (±0.0020)	0.9978 (±0.0020)	0.9908 (±0.0029)	0.9992 (±0.0023)
Quality (norm)	0.7789 (±0.0234)	0.7736 (±0.0190)	0.9638 (±0.0085)	0.9707 (±0.0078)
Idle UAVs	0.0000 (±0.0000)	0.0000 (±0.0000)	0.4333 (±0.5683)	0.0000 (±0.0000)
Sending token	100 (±0.0000)	103.4667 (±1.9070)	104.5333 (±1.8889)	298.1333 (±2.5695)
Cost = (t + me) / ct	1.9660	1.9697	1.1055	2.3811
100 UAVs and 500 tasks in area of 750x750 pixels with deadline of 1000 ticks				
Total reward	227.1356 (±8.4193)	230.5901 (±9.4936)	435.3937 (±5.3410)	461.5992 (±2.0479)
Comp. tasks (norm)	0.7295 (±0.0213)	0.7420 (±0.0230)	1.0000 (±0.0000)	1.0000 (±0.0000)
Elapsed time (norm)	0.9983 (±8.0E-4)	0.9982 (±8.0E-4)	0.9925 (±0.0028)	0.9856 (±0.0123)
Quality (norm)	0.7960 (±0.0173)	0.7902 (±0.0154)	0.9650 (±0.0138)	0.9778 (±0.0058)
Idle UAVs	0.0000 (±0.0000)	0.0000 (±0.0000)	38.7333 (±2.6773)	0.0000 (±0.0000)
Sending token	100 (±0.0000)	110.1667 (±2.6141)	65.7667 (±4.2966)	526.1667 (±12.3821)
Cost = (t + me) / ct	3.0111	2.9875	2.1165	3.0235

The AL algorithm presented similar results to Swarm-GAP, in both scenarios (vii and viii). SAL had an increase of 109.36% and 91.68% in the total reward, compared to Swarm-GAP, in scenarios vii and viii, respectively. LAL also outperformed Swarm-GAP in both scenarios. The total reward had an improvement of 48.72% and 103.22% when LAL is used, compared to Swarm-GAP, respectively in scenarios vii and viii.

Although the LAL method presents the best results in most experiments, it was overcome by SAL in scenario vii. Due to the limitation imposed by the LAL, each UAV can select no more than 3 tasks, in total, in this scenarios. It because the deadline is 300 ticks and there are 100 UAVs (at each tick one UAV receive the token). Thus, always less than 300 tasks will be done, even if there are 500 tasks to perform. On the other hand, in the scenario viii, where there are quite a long time to UAVs perform the tasks (1000 ticks of deadline), the LAL is the best again (in terms of total reward).

7. Related work

Many researchers from the area of multi-agent systems have made efforts to address task allocation problems and several approaches have been proposed. It is outside the scope of this article to present an extension review on the subject. Instead, this section briefly comments on related work that present a more comprehensive literature review.

A review of allocation task can be find in [12]. Nevertheless, the aim of authors in [12] was used Swarm-GAP in a RoboCup Rescue scenario with three kinds of agents. Through experiments the authors showed that Swarm-GAP and LA-DCOP[14] had similar results and they outperformed a greedy strategy.

In [15] the authors have presented a self-organized method for allocating agents to sequentially independent tasks. Furthermore, the authors in [15] also introduce a review that presents work of orchestrated task allocation and self-organized task allocation.

The authors in [16] have presented a multi-objetive optimization for dynamic task allocation. The solution is auction-based. Moreover, in [16] the authors have states a review of several task allocation system, classifying the methods in online-offline, centralized-decentralized and type of interaction.

As in [16], the authors in [17] also used auction-based approach. The aim in [17] is to solve task allocation among UAVs like the present paper. However, this approach is different that one used in present work, which is a approaches based on the nature. The intention here is also not to make a comparison of different approaches. Comparisons between this approaches and communication way were studied in [18, 19]

8. Conclusion

This paper has proposed a solution to the task allocation problem in a team of UAVs in a decentralized way. The tackled problem assumes that the tasks are created by a central entity, such as in several military operations, and refer to the activity of monitoring an area with the objective of detecting a certain type of target.

The proposed solution was developed from the Swarm-GAP algorithm, which use a swarm intelligence approach based on response threshold model. Through experiments observations, features were identified that could promote better allocation of tasks, and consequently increases the total reward. The features were begin used to evolve the solution, resulting in three algorithm variations: Allocation Loop (AL), Sorting and Allocation Loop (SAL) and Limit and Allocation Loop (LAL). These algorithms were evaluated, presenting positive results compared to Swarm-GAP.

In the experiments, it was assumed that communication among UAVs is complete and never fails. However, in a real environment this situation rarely happens. Thus, an important future work is to consider fault tolerance. Another issue to be addressed is the dependency between tasks and teamwork (e.g., two or more agents are necessary to perform the same task).

References

[1] M. Shirzadeh, H. J. Asl, A. Amirkhani, A. A. Jalali, Vision-based control of a quadrotor utilizing artificial neural networks for tracking of moving targets, Engineering Applications of Artificial Intelligence 58 (2017) 34–48.

[2] X. Sun, C. Cai, J. Yang, X. Shen, Route evaluation for unmanned aerial vehicle based on type-2 fuzzy sets, Engineering Applications of Artificial Intelligence 39 (2015) 132–145.

[3] G. P. Kladis, J. T. Economou, K. Knowles, J. Lauber, T.-M. Guerra, Energy conservation based fuzzy tracking for unmanned aerial vehicle missions under a priori known wind information, Engineering Applications of Artificial Intelligence 24 (2) (2011) 278–294.

[4] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, D. Nakazawa, Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles, Springer Science & Business Media, 2010.

[5] C. Zheng, M. Ding, C. Zhou, L. Li, Coevolving and cooperating path planner for multiple unmanned air vehicles, Engineering Applications of Artificial Intelligence 17 (8) (2004) 887–896.

[6] K. Smith, R. F. Stengel, Autonomous control of uninhabited combat air vehicles in heavily-trafficked military airspace, in: 14th AIAA Aviation Technology, Integration, and Operations Conference, 2014, p. 2287.

[7] B. D. Song, J. Kim, J. Kim, H. Park, J. R. Morrison, D. H. Shim, Persistent uav service: an improved scheduling formulation and prototypes of system components, Journal of Intelligent & Robotic Systems 74 (1-2) (2014) 221–232.

[8] X. Qu, W. Zhang, X. Wang, Research of uavs' attack strategy under uncertain condition, Flight Dynamics 4 (2015) 021.

[9] D. B. Shmoys, É. Tardos, An approximation algorithm for the generalized assignment problem, Mathematical programming 62 (1-3) (1993) 461–474.

[10] P. R. Ferreira Jr, F. S. Boffo, A. L. Bazzan, A swarm based approximated algorithm to the extended generalized assignment problem (e-gap), in: Proceedings of the 6th International Joint Conference on Autonomous Agents And Multiagent Systems (AAMAS), ACM, 2007, pp. 1231–1233.

[11] G. Theraulaz, E. Bonabeau, J. Denuebourg, Response threshold reinforcements and division of labour in insect societies, Proceedings of the Royal Society of London B: Biological Sciences 265 (1393) (1998) 327–332.

[12] P. R. Ferreira Jr, F. Dos Santos, A. L. Bazzan, D. Epstein, S. J. Waskow, Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies, Autonomous Agents and Multi-Agent Systems 20 (3) (2010) 421–443.

[13] S. Tisue, U. Wilensky, Netlogo: A simple environment for modeling complexity, in: International conference on complex systems, Vol. 21, Boston, MA, 2004, pp. 16–21.

[14] P. Scerri, A. Farinelli, S. Okamoto, M. Tambe, Allocating tasks in extreme teams, in: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, ACM, 2005, pp. 727–734.

[15] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, M. Dorigo, Self-organized task allocation to sequentially interdependent tasks in swarm robotics, Autonomous agents and multi-agent systems 28 (1) (2014) 101–125.

[16] A. T. Tolmidis, L. Petrou, Multi-objective optimization for dynamic task allocation in a multi-robot system, Engineering Applications of Artificial Intelligence 26 (5) (2013) 1458–1468.

[17] D. Landén, F. Heintz, P. Doherty, Complex task allocation in mixed-initiative delegation: A uav case study, in: International Conference on Principles and Practice of Multi-Agent Systems, Springer, 2010, pp. 288–303.

[18] N. Kalra, A. Martinoli, Comparative study of market-based and threshold-based task allocation, in: Distributed autonomous robotic systems 7, Springer, 2006, pp. 91–101.

[19] Y. Xu, P. Scerri, K. Sycara, M. Lewis, Comparing market and token-based coordination, in: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, ACM, 2006, pp. 1113–1115.