Engineering Applications of Artificial Intelligence 72 (2018) 10–20

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Solving task allocation problem in multi Unmanned Aerial Vehicles systems using Swarm intelligence

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

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

ARTICLE INFO

Keywords:
Unmanned Aerial Vehicles
Task allocation
Multi-agent systems
Swarm intelligence

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. The assumption that tasks are *created* by a central entity is a reasonable one, given the way strategic planning is carried up in military operations. To enable the UAVs to have the ability to decide which tasks to perform, concepts from swarm intelligence and multi-agent system approach are used. Heuristic methods are commonly used to solve this problem, but they present 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 paper proposes three algorithm variants 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 method was evaluated, yielding enhanced results for the addressed problem compared to existing methods reported in the literature.

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 Shirzadeh et al. (2017), Sun et al. (2015) and Kladis et al. (2011). A special case is the use of UAVs for military purpose (Nonami et al., 2010). New applications with multiple UAVs have been planned (Zheng et al., 2004; Smith and Stengel, 2014; Song et al., 2014; Qu et al., 2015), in which UAVs cooperatively patrol perimeters, monitor areas of interest or escort convoys. Areas of difficult access, borders 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. Examples of these sensors are: image sensors like RGB or thermal cameras, chemical sensors, radar sensors, among others. For instance, RGB cameras can be used in surveillance tasks to identify an object of interest while thermal cameras can be used in search and rescue operations, detection of fire spots or night vision. 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 (Alighanbari and How, 2005).

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

* Corresponding author.
E-mail addresses: jschwarzrock@inf.ufrgs.br (J. Schwarzrock), izacarias@inf.ufrgs.br (I. Zacarias), bazzan@inf.ufrgs.br (A.L.C. Bazzan), ricardo@cds.eb.mil.br (R.Q. de Araujo Fernandes), moreira@cds.eb.mil.br (L.H. Moreira), epfreitas@inf.ufrgs.br (E.P. de Freitas).

https://doi.org/10.1016/j.engappai.2018.03.008
Received 7 April 2017; Received in revised form 24 September 2017; Accepted 12 March 2018

autonomously and cooperatively complete a mission assigned by the central command entity. A 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, in which the UAVs are the agents and the mission is associated with a set of tasks. Many efforts have ever been made to solve the task allocation problem in several domains, and different approaches have been proposed, such as threshold-based (Ferreira Jr. et al., 2007; Scerri et al., 2005; Ferreira Jr et al., 2010; Ikemoto et al., 2010) and market-based methods (Lemaire et al., 2004; Landén et al., 2010; Ibri et al., 2012; Tolmidis and Petrou, 2013). In Scerri et al. (2005), for instance, a threshold-based algorithm was proposed to solve the task allocation in a rescue operation scenario. In Landén et al. (2010), an auction-based method was proposed to solve the multi-agent task allocation in the context of a multi-UAV system. Unlike the present problem, in Scerri et al. (2005) and Landén et al. (2010) the tasks are not sent by a central entity, but they are perceived by the agents in the environment.

Swarm intelligence is an appropriate alternative to deal with the multi-UAV task allocation problem in a decentralized way by using a threshold-based approach. Thus, each UAV can decide which tasks it will perform considering only local information, such as its location and resources status. This problem can be modeled using the generalized assignment problem (GAP). The GAP is known to be NP-Complete (Shmoys and Tardos, 1993). In the related literature, there is a heuristic method for task allocation based on swarm intelligence, called Swarm-GAP (Ferreira Jr. et al., 2007) (see Section 2), 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 when 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 when 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 less 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 before, the assumption of the existence of a central commander 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 the proposal of a new method with three algorithm variants, 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. In Section 2, the Swarm-GAP algorithm is briefly presented. Section 3 states the problem. 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 discusses related work. Section 8 then makes concluding remarks and indicates future directions.

2. Background

The authors in Ferreira Jr. et al. (2007) 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 the agents. The task allocation is modeled as a generalized assignment problem (GAP) and the goal is to maximize the total capability. Swarm-GAP is a probabilistic approach, by using the mathematical response threshold model formulated by Theraulaz et al. in (Theraulaz et al., 1998).

The response threshold $T_{\theta_{ij}}$ (Eq. (1)) expresses the likelihood of an agent to react to task-associated stimuli st_j and thus to execute it. The threshold θ_{ij} is based on the agent's capability to perform a task (Eq. (2)). The higher the capacity, the lower the threshold. In case in which the task has a low-stimulus, it is performed by specialized agents. As the stimulus of a task increases (and hence the just mentioned likelihood), less specialized agents also start performing the task (Bonabeau et al., 1999).

$$T_{\theta_{ij}}(st_j) = \frac{st_j^2}{st_j^2 + \theta_{ij}^2} \tag{1}$$

$$\theta_{ij} = 1 - k_{ij} \tag{2}$$

Swarm-GAP uses a communication model based on a token passing protocol, as can be observed in its pseudo code presented in Algorithm 1. Once 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 $T_{\theta_{ij}}$ (Eq. (1)). This tendency is calculated with the task's stimulus st_j and threshold θ_{ij}.

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_{θ_ij(st)}
6:      if roulette() < T_{θ_ij(st)} and r_i ≥ 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 Ferreira Jr. et al. (2007) or Ferreira Jr et al. (2010).

3. Problem formulation

In the problem addressed in this work, the central command unit (henceforth referred simply as central) creates the missions and sends

them to the teams of UAVs, but it does not control the allocation of tasks within the teams. There is no hierarchy among the UAVs in a team and there is no hierarchy among the teams. In order to allow teams to handle several types of tasks, such teams are formed as heterogeneous as possible. However, it is important to highlight that our proposal is to assign tasks to UAVs that are already deployed, i.e. UAVs that are already part of a team operating over a given region. The formation of the teams as well as the elaboration of the missions, represent other problems that are not in this scope, but they can be handled in future complementary work.

Each team operates in a particular region, i.e. the teams are delimited by the region in which they operate. The missions have to be performed in a given region of interest, and the assignment of the team that will perform them is done by the proximity between the team and the mission location. The allocation of the tasks is done within each team and there is no influence or interference of one team in another regarding the task allocation. Hence, the problem addressed in this work refers to the task allocation inside a single team. A non-dynamic team formation is here considered, although in real situations it is possible to consider that the UAVs of a team may change (a discussion about this aspect is presented in Section 8).

Each task of a mission created by the 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. In this case, the central can send reinforcements to that region, according to the demand of remaining tasks, but this is already another problem which is beyond the scope of this work.

The goal here 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 (Shmoys and Tardos, 1993). The problem formulation is given as follows.

Let $\mathcal{I} = \{i_1, \ldots, i_m\}$ be a set of m UAVs and $\mathcal{S} = \{s_1, \ldots, s_u\}$ 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 $\mathcal{J} = \{j_1, \ldots, j_n\}$ of n tasks and it has 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 $\langle x, y\rangle$ of the task's location; $t \in \mathcal{A} = \{a_1, \ldots, a_v\}$ 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. (3). When $Q(i, j) = 0$ the UAV i no have sensor capable of detecting t_j.

$$Q(i, j) = \max_{s \in Si_i} q_{st_j} \tag{3}$$

Each agent i has a specific capability k_{ij} to perform each task j. The capability in this problem is defined by Eq. (4), 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.

$$k_{ij} = \frac{\max_{g\in J}\{d(i,g)\} - d(i,j)}{\max_{g\in J}\{d(i,g)\}} \times \alpha + \left(1 - \frac{\max_{g\in J}\{Q(i,g)\} - Q(i,j)}{\max_{g\in J}\{Q(i,g)\}}\right) \times (1 - \alpha) \tag{4}$$

According to Eq. (4), 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. (5)),

$$X = \underset{X'}{\operatorname{argmax}} \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} k_{ij} \times x'_{ij} \tag{5}$$

subject to constraints:

$$\forall j \in \mathcal{J} \sum_{i\in\mathcal{I}} x_{ij} \leq 1 \tag{6}$$

$$\forall x \in X \mid x_{ij} = 1, Q(i, j) > 0 \tag{7}$$

$$\forall i \in \mathcal{I} \sum_{j\in\mathcal{J}} x_{ij} \times C_{ij} \leq r_i \tag{8}$$

The constraint in Eq. (6) ensures that each task is allocated to at most one UAV. Eq. (7) restricts the allocation of the tasks only to the agents that have the quality to execute them. Finally, the constraint in Eq. (8) requires that the UAV's resources limitations must be respected.

For simplicity, a single type of resource is being considered: 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.

4. Proposed solution

The proposal based on Swarm-GAP presented in this work is divided in three variants. This section describes each of them as well as the motivation behind them.

4.1. Allocation Loop (AL)

When Swarm-GAP is used to solve the task allocation problem presented in Section 3, 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 do not 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 is or is not available 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.

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
```

4.2. Sorting and Allocation Loop (SAL)

In the previous AL algorithm, 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 tasks by the UAVs, i.e., it makes easier to the UAVs select the more 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_θij
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
```

4.3. 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) algorithm 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.

Nevertheless, since the method is probabilistic, there is no guarantee that all agents will select some task. The method does not enforce an agent to choose a task; it only limits the number of tasks that are selected per round. Therefore, there may still be cases with idle agents. However, in general, the workload balance improves. Moreover, cases in which an optimal allocation of tasks is achieved may happen even when there are some idle agents.

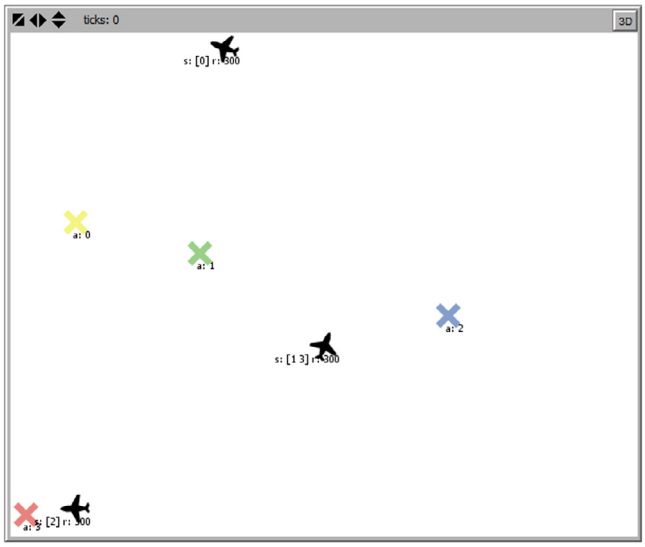

Fig. 1. Simulation in the NetLogo.

5. Experimental setup

In order to evaluate the proposed solutions to the task allocation problem among UAVs, a simulation was implemented in NetLogo (Tisue and Wilensky, 2004) (version 5.3.1). The tasks and UAVs are represented by specific symbols. Fig. 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.

According to problem formulation (Section 3), each task has its location $\langle x, y \rangle$, 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 in Fig. 1, for example, the color red may represent a fire places and the color blue a train of parked vehicles. Each UAV, in addition to its location $\langle x, y \rangle$ 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 (simulation time progression unity), 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 perform. It is noteworthy that the makespan (elapsed time) may be easily converted 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 receives the token adds its ID in the list. At each tick, the token is sent to a random UAV that has not yet received the token in the current round, i.e. a UAV that is not in the list of visited agents. The token also has a list of unavailable agents. This list is used by the proposed algorithms AL, SAL and LAL (see Section 4.1). When a UAV receives the token, it runs the algorithm to choose the tasks it will perform.

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

Table 1
Quality of each sensor to detect the different 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

(i) 3 UAVs; 4 tasks; 300 ticks as deadline; 100 × 80 px area size.
(ii) 3 UAVs; 8 tasks; 300 ticks as deadline; 100 × 80 px area size.
(iii) 3 UAVs; 16 tasks; 300 ticks as deadline; 100 × 80 px area size.
(iv) 3 UAVs; 32 tasks; 300 ticks as deadline; 100 × 80 px area size.
(v) 6 UAVs; 64 tasks; 300 ticks as deadline; 200 × 160 px area size.
(vi) 9 UAVs; 96 tasks; 300 ticks as deadline; 300 × 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 $\langle x, y\rangle$ were defined randomly, 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 $\langle x, y\rangle$ and the type of target were defined randomly. 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 tasks.

Table 1 presents the sensors' quality to detect each type of target. These 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, while sensors s_1 and s_3 are unable to detect a_0.

The scenarios were run 30 times for each proposed algorithm. In all experiments, the UAVs' capabilities were computed using Eq. (4) with $\alpha = 0.6$, giving higher 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, as it is known for the Swarm-GAP (Ferreira Jr et al., 2010).

The experiments were conducted in a PC with 1.70 GHz, 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 algorithms, the following metrics were considered: (a) Total reward; (b) Makespan (total elapsed time from the beginning to the end of the execution); (c) Quantity of the tasks that were completed; and (d) Quality of the completed tasks. The main objective is to maximize the total reward (a). Since this metric is influenced by the quantity and quality of the completed tasks (see Section 3), the metrics (b), (c) and (d) allow to understand the result of the (a). Furthermore, the (e) Number of exchanged messages (related to the token-passing mechanism) and the (f) Algorithm's runtime were also measured and analyzed.

The metrics are applied to evaluate the proposed methods, i.e. AL, SAL and LAL, and compared them to the results obtained using the Swarm-GAP. Part of the achieved results are shown in charts, but the complete data (mean and standard deviation) for all assessed metrics for each algorithm and scenario, from (i) to (vi), are presented in Tables 2 and 3. A specific set of scenarios is used to highlight the results for each metric. After their presentation, these results are discussed. At the end of this section, a scalability analysis of the algorithms is provided.

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 the completed tasks and number of exchanged messages. These four scenarios were selected due to the fact that they have the same number of UAVs and the amount of tasks is varied, but with the same mission's deadline. This choice allows to evaluate situations in which the UAVs have enough time, or more than enough, to perform the tasks (this is the case for scenarios (i) and (ii)), and situations in which the UAVs have a short time to perform a large number of tasks (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 the different scenarios is shown in Fig. 2. In scenarios with 4 and 8 tasks, both AL and SAL algorithms outperformed the Swarm-GAP by about 24%.

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 in the scenario with 32 tasks.

The highest total reward was obtained by the LAL algorithm. This variant outperformed Swarm-GAP by 31.85%, 64.63%, 59.2% and 118.7% in the scenarios with 4, 8, 16 and 32 tasks, respectively. This pattern of enhancement in the total reward can also be observed in the scenario with 64 and 96 tasks (See Table 3). The LAL algorithm presented results 3.2 and 2.87 times better than Swarm-GAP.

6.2. Makespan and quantity of completed tasks

The quantity of completed tasks for each algorithm in each scenario is presented in Fig. 3. The results presented in this figure are normalized by the number of tasks of each scenario. Fig. 4 shows the elapsed time (in ticks), normalized by the deadline associated to the mission. As it is possible to observe, using time efficiently, the UAVs can perform more tasks achieving greater total rewards.

In scenarios with 4 and 8 tasks, in which there is enough time (extended or relaxed deadline) for the UAVs carrying out all the tasks (the whole mission), the Swarm-GAP algorithm results in UAVs not using all the available resources. In these cases they have used only 35.59% and 68.81% of the time. 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, using Allocation Loop (AL) method allowed 100% and 99% of the tasks to be performed. This was possible because while a UAV still has resources, it may select more tasks to perform in this method.

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 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 scenarios with 16 and 32 tasks, i.e., those in which 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, it happened simulation runs in which 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.

Table 2
Total reward, makespan, quantity and quality of the completed tasks, number of exchanged messages and algorithm's runtime for 30 runs of each algorithm in scenarios with 3 UAVs and different number of tasks.

	Swarm-GAP Mean (St.Dev.)	AL Mean (St.Dev.)	SAL Mean (St.Dev.)	LAL Mean (St.Dev.)
3 UAVs and 4 tasks in area of 100 × 80 pixels with deadline of 300 ticks				
Total reward	1.6971 (±0.4705)	2.1016 (±0.3251)	2.0947 (±0.3579)	2.2377 (±0.3217)
Elapsed time (norm)	0.3559 (±0.1874)	0.431 (±0.1388)	0.4139 (±0.1486)	0.3118 (±0.0759)
Comp. tasks (norm)	0.7583 (±0.1796)	1.0000 (±0.0000)	1.0000 (±0.0000)	1.0000 (±0.0000)
Quality (norm)	0.8131 (±0.1784)	0.8242 (±0.1284)	0.7942 (±0.1488)	0.9167 (±0.1428)
Sending token	2.8667 (±0.3457)	4.2000 (±1.7889)	4.0333 (±1.7317)	6.9667 (±2.2816)
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 100 × 80 pixels with deadline of 300 ticks				
Total reward	2.9436 (±0.4874)	3.651 (±0.2064)	3.6006 (±0.3246)	4.8462 (±1.3581)
Elapsed time (norm)	0.6881 (±0.1722)	0.7731 (±0.1103)	0.7382 (±0.1264)	0.6786 (±0.0676)
Comp. tasks (norm)	0.7708 (±0.1316)	0.9958 (±0.0228)	0.9833 (±0.0432)	1.0000 (±0.0000)
Quality (norm)	0.7502 (±0.1599)	0.7699 (±0.0892)	0.7273 (±0.1003)	0.8125 (±0.1247)
Sending token	2.9667 (±0.1826)	6.8000 (±2.6182)	6.0000 (±2.3781)	11.1667 (±2.3057)
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 100 × 80 pixels with deadline of 300 ticks				
Total reward	6.6424 (±1.0648)	7.6047 (±0.8502)	8.7607 (±0.6316)	10.5753 (±0.4666)
Elapsed time (norm)	0.9179 (±0.0495)	0.9462 (±0.0354)	0.9210 (±0.0496)	0.9027 (±0.0599)
Comp. tasks (norm)	0.7229 (±0.0815)	0.8604 (±0.0585)	0.9313 (±0.0474)	0.9896 (±0.0288)
Quality (norm)	0.8110 (±0.1040)	0.7431 (±0.0990)	0.7849 (±0.0591)	0.9239 (±0.0588)
Sending token	3.0000 (±0.0000)	5.7333 (±2.5316)	6.3667 (±1.4259)	18.7667 (±1.9420)
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 100 × 80 pixels with deadline of 300 ticks				
Total reward	9.1017 (±1.4553)	9.0073 (±1.2457)	13.7469 (±0.8765)	19.9057 (±0.4189)
Elapsed time (norm)	0.9624 (±0.0177)	0.9652 (±0.0185)	0.9554 (±0.0168)	0.9422 (±0.0186)
Comp. tasks (norm)	0.4781 (±0.0533)	0.4844 (±0.0498)	0.6094 (±0.0391)	0.7604 (±0.0222)
Quality (norm)	0.7416 (±0.0723)	0.7275 (±0.0910)	0.8741 (±0.0512)	0.9315 (±0.0292)
Sending token	3.0000 (±0.0000)	3.5333 (±0.8996)	3.8333 (±0.9129)	24.8000 (±1.4716)
Total runtime (sec)	2.9778 (±0.7843)	3.2434 (±1.1824)	4.0268 (±1.0639)	10.9934 (±2.1070)

Table 3
Total reward, makespan, quantity and quality of the completed tasks, number of exchanged messages and algorithm's runtime of 30 runs of each algorithm in scenarios with different number of UAVs and tasks.

	Swarm-GAP Mean (St.Dev.)	AL Mean (St.Dev.)	SAL Mean (St.Dev.)	LAL Mean (St.Dev.)
6 UAVs and 64 tasks in area of 200 × 160 pixels with deadline of 300 ticks				
Total reward	12.1152 (±1.9136)	12.9234 (±1.8407)	28.2929 (±1.0694)	38.7922 (±1.2800)
Elapsed time (norm)	0.9850 (±0.0086)	0.9819 (±0.0100)	0.9737 (±0.0101)	0.9643 (±0.0116)
Comp. tasks (norm)	0.3135 (±0.0323)	0.3302 (±0.0317)	0.5271 (±0.0201)	0.6813 (±0.0208)
Quality (norm)	0.7358 (±0.0790)	0.7447 (±0.0689)	0.9582 (±0.0319)	0.9667 (±0.0165)
Sending token	6.0000 (±0.0000)	6.5333 (±0.9371)	6.6333 (±1.0334)	44.0000 (±1.5086)
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 300 × 240 pixels with deadline of 300 ticks				
Total reward	15.582 (±2.0050)	16.724 (±2.1908)	37.9608 (±1.1119)	44.733 (±1.5961)
Elapsed time (norm)	0.9886 (±0.0082)	0.9894 (±0.0064)	0.9763 (±0.0092)	0.9693 (±0.0124)
Comp. tasks (norm)	0.2611 (±0.0217)	0.2774 (±0.0276)	0.4674 (±0.0170)	0.5226 (±0.0162)
Quality (norm)	0.7899 (±0.0599)	0.7865 (±0.0641)	0.9680 (±0.0202)	0.9752 (±0.0159)
Sending token	9.0000 (±0.0000)	9.8667 (±1.1958)	9.7000 (±1.2360)	51.500 (±1.4797)
Total runtime (sec)	9.0990 (±4.6221)	13.1253 (±7.6605)	12.2062 (±2.7515)	50.5544 (±31.2161)

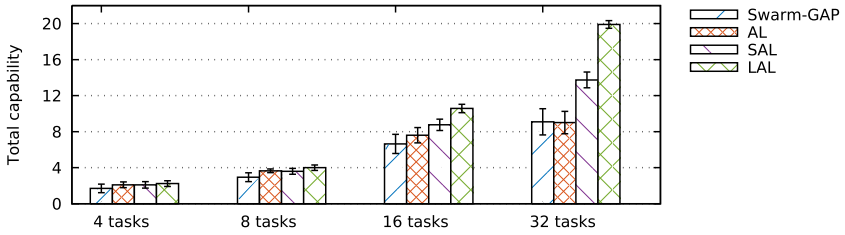

Fig. 2. Total reward of each method for different scenarios with stimulus 0.6.

6.3. Quality of completed tasks

The average quality of the completed tasks in each scenario is shown in Fig. 5. The quality metric was normalized in relation to the number of completed tasks. The higher the quality of the tasks performed by the agents, the greater the total reward. For this measure, the LAL algorithm also shown the best results.

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

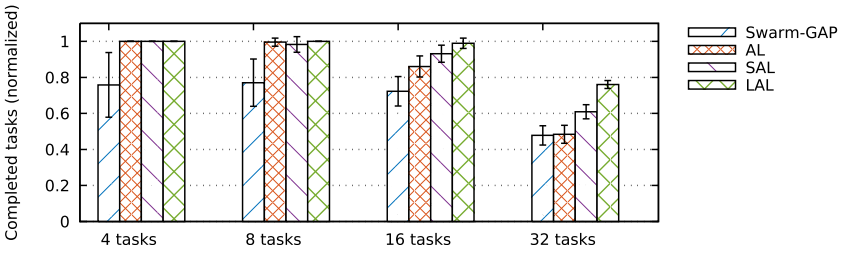

Fig. 3. Completed tasks of each method for different scenarios with stimulus 0.6.

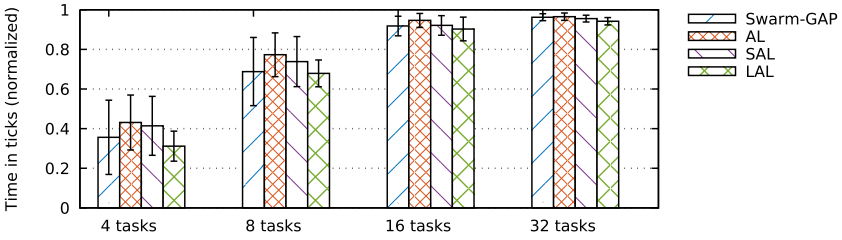

Fig. 4. Makespan of each method for different scenarios with stimulus 0.6.

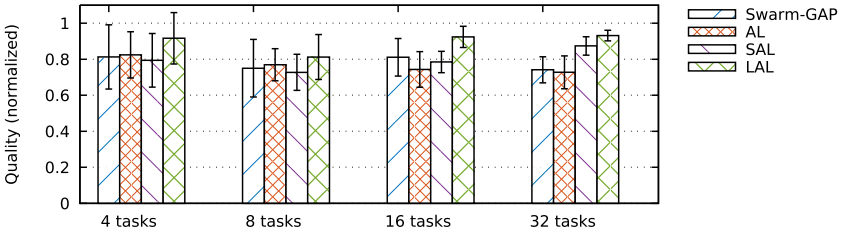

Fig. 5. Quality of each method for different scenarios with stimulus 0.6.

In the scenario with 16 tasks, SAL 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, SAL improved the quality by 17.86% compared to the Swarm-GAP. In the scenarios with 16 and 32 tasks, LAL presents an improvement of 13.92% and 25.6% compared to the Swarm-GAP, respectively. 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

Fig. 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 the tokens that are sent relates more to the amount of agents than to the number of performed tasks.

Using Swarm-GAP, each UAV receives the token at most once. When the UAV does not receive it, this means that all tasks were already selected before the token arrives to this given UAV. Anyway, when there is a given number of agents, for example three, the token is sent three times and then it is discarded. 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 choosing at most one task at a time that the UAV receives the token, the number of exchanged messages is about the same number of performed tasks.

Despite the higher 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 investigate if this increase in the number of exchanged messages is rewarded by the quantity of completed tasks, Fig. 7 presents the average cost to perform each task, considering also the number of exchanged messages. Then, the cost was computed by $avgcost = (t + em)/ct$, where (t) is the elapsed time, (em) is number of exchanged messages and (ct) is the amount of completed tasks.

Even assuming that the amount of exchanged messages is also considered in the cost of executing the tasks, 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

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

The results of 30 runs (mean and standard deviation) of these scenarios, with all considered metrics (see Section 6), can be seen in Tables 2 and 3. Here, the runtime analysis of each algorithm in different scenarios is highlighted in 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. It is worth to notice that the number of times the algorithm was executed is the same of the number of exchanged messages.

The total runtime for AL and SAL are almost the same for each scenario. This suggests that sorting the tasks in SAL algorithm has not a

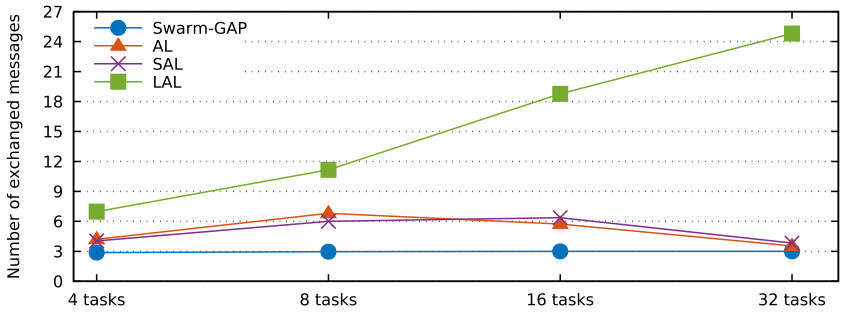

Fig. 6. Total number of exchanged messages of each method for different scenarios with stimulus 0.6.

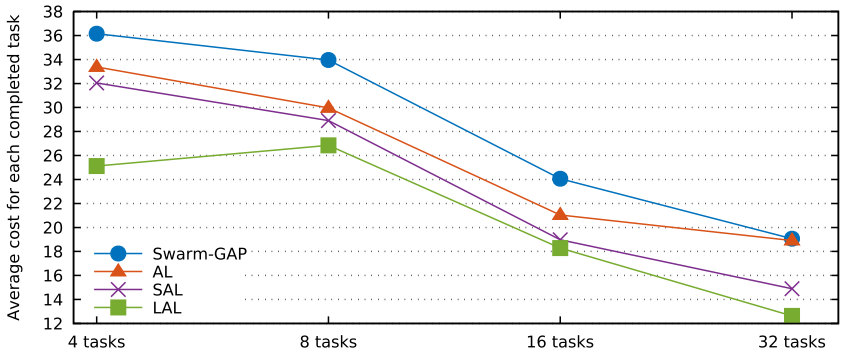

Fig. 7. Average cost of the completed tasks (considering number of exchanged messages) of each method for different scenarios with stimulus 0.6.

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

great impact in the runtime. The AL and SAL results are close to Swarm-GAP in the scenario 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 for all scenarios compared to the others algorithms. Due to the limit to select tasks, 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 s, on average, to execute the algorithm each time it received the token in this scenario with 96 tasks.

6.6. Discussions

The AL algorithm allows the UAVs to perform more tasks than Swarm-GAP by enabling the UAVs to select tasks while they still have resources to accomplish them. Consequently, the elapsed time increases, since more tasks are performed. This is a positive feature of the proposed approach because the UAVs can use all available time. Furthermore, the number of exchanged messages among the UAVs, does not increase significantly when using AL. 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 cannot perform any of the tasks that are still available in the token. However, as said, 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, a UAV may end up using its time to perform other, less appropriate tasks. Thus, more tasks are performed than when using AL (without increasing the elapsed time and with higher quality). This provides 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 the tasks selection limit. Thus, it improves performance further than AL and SAL. More tasks are performed in less time and the completed tasks quality increased even more. When using LAL, each UAV can choose just one task at a time, preventing some UAVs from selecting many tasks, becoming overloaded, while others UAVs remain idle. However, the possibility of reaching an optimal allocation keeping some agents in idle is not ruled out. Such situations may happen. That is why the LAL method does not force an agent to choose a task: it can or cannot select a task, but only one at the most.

By construction, LAL causes more exchanged messages among the UAVs. Nevertheless, it is counter-balanced by the amount of performed

tasks 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 the 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) and differ from each other by the missions' deadline. These two additional scenarios are described as follows:

(vii) 100 UAVs; 500 tasks; 300 ticks as deadline; 750 × 750 px area size.
(viii) 100 UAVs; 500 tasks; 1000 ticks as deadline; 750 × 750 px area size.

It is worth mentioning that the quantities used in these scenarios extrapolate those that would be expected in a real situation. However, the intention here is to stress these numbers to discuss scalability. The results (total reward, makespan, quantity and quality of completed tasks and number of exchanged messages) of 30 runs are shown in Table 5.

The AL algorithm presented similar results to Swarm-GAP, in both scenarios (vii) and (viii). SAL had an increase of around 100% in the total reward, compared to Swarm-GAP, in both scenarios. 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 of the performed experiments, it was overcome by SAL in scenario (vii). Due to the limit of selection imposed by the LAL, each UAV can select no more than 3 tasks in total. This is because the deadline is 300 ticks and there are 100 UAVs (at each tick one UAV receives the token). Thus, always less than 300 tasks will be performed, even if there are 500 tasks to be performed. On the other hand, in the scenario (viii), in which there is a large time window to the UAVs perform the tasks (1000 ticks of deadline), LAL is the best 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, hence several approaches have been proposed. This section discusses some of these proposal that deal with this problem in different domains as well as in multi-UAV systems. The focus will be on decentralized approaches. Although there are also centralized solutions, such as Jose and Pratihar (2016) which uses genetic algorithm to assign tasks in a centralized multi-robots system for industrial plant inspection, such solutions are not suitable for the problem addressed in this current work.

A number of proposals employed the threshold-based approach to solve task allocation problems, such as Scerri et al. (2005), Ferreira Jr et al. (2010) and Ikemoto et al. (2010) and Swarm-GAP (Ferreira Jr. et al., 2007). The authors in Scerri et al. (2005) proposed the LA-DCOP, an distributed threshold-based algorithm to task allocation in rescue scenarios with extreme teams of agents. In Ferreira Jr et al. (2010) the authors showed that Swarm-GAP (Ferreira Jr. et al., 2007) and LA-DCOP (Scerri et al., 2005) had similar results in a RoboCup Rescue scenario with three kinds of agents. In Ikemoto et al. (2010), the threshold-based approach is applied to food forage labor by a group of robots.

The approach used in this paper is the threshold-based as in Ferreira Jr. et al. (2007), Scerri et al. (2005), Ferreira Jr et al. (2010) and Ikemoto et al. (2010) and the problem presented here also deals with extreme teams as in Scerri et al. (2005) and Ferreira Jr et al. (2010) due to the different types of sensors that the UAVs have. However, the way that the agents receive the tasks is different. The tasks in the addressed problem are ordered by central entity, due to strategic military purposes. This generates a greater amount of tasks received by the agents in a single time compared to the assumption that the agents perceive the tasks in the environment as in Ferreira Jr. et al. (2007), Scerri et al. (2005), Ferreira Jr et al. (2010) and Ikemoto et al. (2010). It causes drawbacks as discussed in Section 1. However, it is worth to highlight that the perception and creation of tasks can, if necessary, be easily added to our method, since it is based on Swarm-GAP that has this functionality.

Market-based approach also have been widely used to deal with the task allocation problem in different domains. Examples of this approach are those works employing auctions in their methods, as such Lemaire et al. (2004), Landén et al. (2010), Ibri et al. (2012) and Tolmidis and Petrou (2013). The authors in Lemaire et al. (2004) and Landén et al. (2010) have applied the task allocation problem in multi-UAV systems as this work. In Lemaire et al. (2004) the authors have stated the problem as a TSP and proposed an algorithm based on the Contract-Net protocol. A token-ring based approach is also used to avoid several auctions being launched at the same time. In Landén et al. (2010), the aim is to solve complex task allocation among UAVs. To deal with complex tasks the authors have proposed a tree-shaped modeling for the problem, allowing to express the hierarchy of dependence among the tasks. The authors in Ibri et al. (2012) used auction to assign task to emergency vehicles. Unlike our work, the fleet is composed of homogeneous vehicles. In Tolmidis and Petrou (2013) the authors proposed a general task allocation solution for multi-robot systems, begin independent of the domain. The solution works with multi-objective optimization. The authors consider their method "neither strictly centralized, nor decentralized" (Tolmidis and Petrou, 2013).

The methods proposed by Lemaire et al. (2004), Landén et al. (2010), Ibri et al. (2012) and Tolmidis and Petrou (2013) are market-based being differently to the one here presented, which uses threshold-based and token-passing approaches, and being also biologically inspired. Comparisons among different approaches to solve the task allocation problem can also be found in Kalra and Martinoli (2006) and Xu et al. (2006). These studies compare market-based, threshold-based and token-passing based approaches, providing insights of their benefits and drawbacks considering different scenarios. Furthermore, in Lemaire et al. (2004), Landén et al. (2010) and Tolmidis and Petrou (2013) the agents are also responsible for perceiving the tasks in the environment as in Ferreira Jr. et al. (2007), Scerri et al. (2005), Ferreira Jr et al. (2010) and Ikemoto et al. (2010). Once an agent perceives a task, it launches the task to the other agents through auction.

There are also other works that propose methods for allocating tasks, such as Iijima et al. (2016, 2017), Branisso et al. (2013) and Brutschy et al. (2014). The authors in Iijima et al. (2016) and Iijima et al. (2017) deal with task allocation in a distributed environment such as the Internet. This assignment is based on the preferences declared by the agents, being closer to an auction-based approach. In Branisso et al. (2013) the authors propose a solution to task allocation in the domain of material handling in warehouses. In this work, tasks should be performed by Automated Guided Vehicles (AGVs). Their solution uses a fuzzy inference system to support decision making. Like our proposal, in their approach, each agent decides by itself which task it will perform. An agent makes the decision based on its location, loading and storage point information. In this problem, the AGVs are homogeneous agents that have to deal with two types of tasks: that are related to loading and storage. On the other hand, the proposal presented in this paper addresses heterogeneous agents (UAVs), which aim to handle a number of different types of tasks. The authors in Brutschy et al. (2014) deal with tasks that are sequentially interdependent in a multi-robot system. Each agent also decides by itself if it performs or not a task. An agent uses its perception of the delay experienced when waiting for other agents working on the other subtask(s) to take a decision. The method requires no communication between agents. The tasks in their problem can be perceived in the environment, unlike the tasks referred to the current

Table 5
Total reward, makespan, quantity and quality of completed tasks and number of exchanged messages of 30 runs of each algorithm in scenarios with 100 UAVs and 500 tasks.

	Swarm-GAP Mean (St.Dev.)	AL Mean (St.Dev.)	SAL Mean (St.Dev.)	LAL Mean (St.Dev.)
100 UAVs and 500 tasks in area of 750 × 750 pixels with deadline of 300 ticks				
Total reward	162.0141 (±6.5974)	162.9333 (±7.0324)	339.2015 (±7.5146)	240.956 (±3.6869)
Elapsed time (norm)	0.9979 (±0.0020)	0.9978 (±0.0020)	0.9908 (±0.0029)	0.9992 (±0.0023)
Comp. tasks (norm)	0.4063 (±0.0141)	0.4090 (±0.0161)	0.7269 (±0.0159)	0.5022 (±0.0075)
Quality (norm)	0.7789 (±0.0234)	0.7736 (±0.0190)	0.9638 (±0.0085)	0.9707 (±0.0078)
Sending token	100 (±0.0000)	103.4667 (±1.9070)	104.5333 (±1.8889)	298.1333 (±2.5695)
100 UAVs and 500 tasks in area of 750 × 750 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)
Sending token	100 (±0.0000)	110.1667 (±2.6141)	65.7667 (±4.2966)	526.1667 (±12.3821)

paper, which are given by a central entity. In summary, despite the similarities in some aspects, the proposals in Branisso et al. (2013) and Brutschy et al. (2014) have different scope and assumption to our, due to the differences in the agents' behavior, the missions that they have to perform and the different assumptions about the agents' characteristics.

The problem in Alighanbari and How (2005) refers to decentralized task assignment for a fleet of UAVs, being more similar to one in this current paper. The UAVs need to visit a set of targets, previously defined. The UAVs must decide which will visit each target, in a decentralized way. The authors in Alighanbari and How (2005) have demonstrated the drawback of using implicit coordination, which the central assignment algorithm is replicated in each UAV. When using implicit coordination all UAVs must have the same situational awareness. Then, a more robust extension is proposed. However their approach still assumes some degree of data synchronization among the UAVs. Even if there is a central entity, which creates the tasks, the proposed algorithm in this paper is purely decentralized, unlike Alighanbari and How (2005) and Tolmidis and Petrou (2013). The central and even the agents do not need to have any kind of information about the others.

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 refers to the activity of monitoring an area with the objective of detecting certain types of targets.

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

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

Experiments with dynamic teams will also be conducted in the future. Although the experiments were performed with non-dynamic teams, the proposed method is flexible enough to support also dynamic assignment of the UAVs to the teams, which is a promising direction for future work. This is possible because agents do not have information on the overall status of the team or information about any other agent. Thus, agents may leave or join the team without influencing the decision-making process.

References

Alighanbari, M., How, J.P., 2005. Decentralized task assignment for unmanned aerial vehicles. In: Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on. IEEE, pp. 5668–5673.

Bonabeau, E., Dorigo, M., Theraulaz, G., 1999. Swarm Intelligence: From Natural to Artificial Systems. Oxford university press.

Branisso, L.B., Kato, E.R.R., Pedrino, E.C., Morandin, O., Tsunaki, R.H., 2013. A multi-agent system using fuzzy logic to increase AGV fleet performance in warehouses. In: Computing Systems Engineering, SBESC, 2013 III Brazilian Symposium on. IEEE, pp. 137–142.

Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., Dorigo, M., 2014. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. Autonom. Agents Multi-Agent Syst. 28 (1), 101–125.

Ferreira Jr., P.R., Boffo, F.S., Bazzan, A.L., 2007. 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, pp. 1231–1233.

Ferreira Jr., P.R., Dos Santos, F., Bazzan, A.L., Epstein, D., Waskow, S.J., 2010. RoboCup Rescue as multiagent task allocation among teams: Experiments with task interdependencies. Autonom. Agents Multi-Agent Syst. 20 (3), 421–443.

Ibri, S., Nourelfath, M., Drias, H., 2012. A multi-agent approach for integrated emergency vehicle dispatching and covering problem. Eng. Appl. Artif. Intell. 25 (3), 554–565.

Iijima, N., Hayano, M., Sugiyama, A., Sugawara, T., 2016. Analysis of task allocation based on social utility and incompatible individual preference. In: Technologies and Applications of Artificial Intelligence, TAAI, 2016 Conference on. IEEE, pp. 24–31.

Iijima, N., Sugiyama, A., Hayano, M., Sugawara, T., 2017. Adaptive task allocation based on social utility and individual preference in distributed environments. Procedia Comput. Sci. 112, 91–98.

Ikemoto, Y., Miura, T., Asama, H., 2010. Adaptive division-of-labor control algorithm for multi-robot systems. J. Robot. Mechatron. 22 (4), 514.

Jose, K., Pratihar, D.K., 2016. Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. Robot. Auton. Syst. 80, 34–42.

Kalra, N., Martinoli, A., 2006. Comparative study of market-based and threshold-based task allocation. In: Distributed Autonomous Robotic Systems 7. Springer, pp. 91–101.

Kladis, G.P., Economou, J.T., Knowles, K., Lauber, J., Guerra, T.-M., 2011. Energy conservation based fuzzy tracking for unmanned aerial vehicle missions under a priori known wind information. Eng. Appl. Artif. Intell. 24 (2), 278–294.

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

Lemaire, T., Alami, R., Lacroix, S., 2004. A distributed tasks allocation scheme in multi-UAV context. In: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, Vol. 4. IEEE, pp. 3622–3627.

Nonami, K., Kendoul, F., Suzuki, S., Wang, W., Nakazawa, D., 2010. Autonomous flying robots: Unmanned aerial vehicles and micro aerial vehicles. Springer Science &Business Media.

Qu, X., Zhang, W., Wang, X., 2015. Research of UAVs' attack strategy under uncertain condition. Flight Dyn. 4, 021.

Scerri, P., Farinelli, A., Okamoto, S., Tambe, M., 2005. Allocating tasks in extreme teams. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, pp. 727–734.

Shirzadeh, M., Asl, H.J., Amirkhani, A., Jalali, A.A., 2017. Vision-based control of a quadrotor utilizing artificial neural networks for tracking of moving targets. Eng. Appl. Artif. Intell. 58, 34–48.

Shmoys, D.B., Tardos, É., 1993. An approximation algorithm for the generalized assignment problem. Math. Program. 62 (1–3), 461–474.

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

Song, B.D., Kim, J., Kim, J., Park, H., Morrison, J.R., Shim, D.H., 2014. Persistent UAV service: An improved scheduling formulation and prototypes of system components. J. Intell. Robot. Syst. 74 (1–2), 221–232.

Sun, X., Cai, C., Yang, J., Shen, X., 2015. Route evaluation for unmanned aerial vehicle based on type-2 fuzzy sets. Eng. Appl. Artif. Intell. 39, 132–145.

Theraulaz, G., Bonabeau, E., Denuebourg, J., 1998. Response threshold reinforcements and division of labour in insect societies. Proc. R. Soc. B 265 (1393), 327–332.

Tisue, S., Wilensky, U., 2004. Netlogo: A simple environment for modeling complexity. In: International Conference on Complex Systems, Vol. 21. Boston, MA, pp. 16–21.

Tolmidis, A.T., Petrou, L., 2013. Multi-objective optimization for dynamic task allocation in a multi-robot system. Eng. Appl. Artif. Intell. 26 (5), 1458–1468.

Xu, Y., Scerri, P., Sycara, K., Lewis, M., 2006. Comparing market and token-based coordination. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, pp. 1113–1115.

Zheng, C., Ding, M., Zhou, C., Li, L., 2004. Coevolving and cooperating path planner for multiple unmanned air vehicles. Eng. Appl. Artif. Intell. 17 (8), 887–896.