A Distributed Control of Movements and Fuzzy Logic-Based Task Allocation for a Swarm of Autonomous Agents

Pedro Lucas, Kleber Loayza and Enrique Peláez
ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL,
Electrical and Computer Engineering Department, Centro de Tecnologías de Información,
Campus Gustavo Galindo Km.30.5 Vía Perimetral, P.O. Box 09-01-5863,
Guayaquil, Ecuador
Email: {pedro.lucas, kloayza, epelaez}@cti.espol.edu.ec

***Abstract*—This paper proposes a decentralized method for controlling a swarm of autonomous agents represented as Unmanned Aerial Vehicles (UAVs) to accomplish a set of tasks cooperatively. The task commissioning is carried out by a Fuzzy Logic-based task allocation strategy, and the movement behavior is based on the internal state of the agent as well as the external data from its neighbors through local communications. The solution was modified from a centralized strategy developed in a previous work in order to be enhanced, so the proposed distributed strategy was tested considering the same experiments, configurations and conditions applied to the centralized approach in a simulated environment. To test the strategy, the planned activity to be performed was planting seeds, in a field composed by a grid of points that represents the places to be sown. Completion times and collision avoidance attempts were measured to test the effectiveness, as well as to perform a comparison between the distributed and centralized methods. Important improvements were found considering the size of agents in a swarm.**

***Keywords*—swarm intelligence, fuzzy logic, agents behavior, task allocation, farming automation, UAVs**

I. Introduction

There is an increasing number of applications for *Unmanned Aerial Vehicles* (UAVs) in recent years, specially for performing complex activities considering domains in which a swarm behavior provides flexibility and robustness; in particular, hard to solve problems where cooperative strategies can make the difference [1]. The global intelligence, which emerges from a group of simple and basic agents, is inspired by nature, which has also led researchers to propose models to reproduce community behaviors in order to solve complex tasks [2].

In a previous work [3] a centralized architecture for movement control and collision avoidance was proposed. *Sowing seeds*, as an activity for precision agriculture, was implemented using this architecture, thus movement coordination and task assignment to accomplish a common objective were designed for a swarm of UAVs. However, as a centralized approach, a complete agent independence is not part of the architecture.

The technique proposed in this research addresses two problems: the *decentralization* of swarm behavior, hence the entire logic for solving simple tasks resides within the agents, leading to a result of collaborative performance through local communication channels; and, the *task allocation* algorithm, which is based on fuzzy logic, hence the time for accomplishing the task is reduced, as well as the number of collision attempts; both are important parameters for saving resources.

Our goal is to test the effectiveness, scalability and stability of the proposed solution as well as to perform a comparison between swarm strategies (centralized and fuzzy-decentralized), considering the same experiments and conditions set in [3].

A. Related Work

There have been different strategies to control and coordinate swarms of individuals. Algorithms such as *Particle Swarm Optimization* (PSO) and *Ant Colony Optimization* (ACO) have been used for such purposes as in [4], [5] and [6]; also, other techniques have been applied for achieving a cooperative behavior such as *Artificial Potential Functions* (APFs) in [7], *Time-Varying Formation* (TVF) [8], and *Fuzzy Logic* as a control method [5] [9] [10]. Nevertheless, these and other proposals for swarm coordination use hybrid algorithms for effectiveness in agent movements or optimization for task completion.

Researchers have been focused on decentralized solutions; in these cases, distributed agents behaviors, considering local communication, have been designed. Wang *et al* [8] developed a protocol to switch swarm topologies formation using state of neighbors; De Souza *et al* [11] addressed the problem of wide-area communications for coordinating UAVs. Other methods considered simple exchange of data on local communication as in [12] and [13]. These studies show the importance of awareness among agents in distributed scenarios.

Usually, a *task allocation* solution is required for swarm activities, as well as for other science applications, in which efficient methods could enhance the performance of a system [14]. For swarm intelligence problems, strategies such as *hybrids* with *PSO* and *Fuzzy Logic* for scheduling [15], *agent negotiation* [16], and *Multi-Criteria Decision Making* (MCDM) [17], have been used. For other applications that consider distributed systems, there have been methods like *distributed polynomial* in [18], and pure *Fuzzy Logic* [14]. The mentioned methods have been proved to be effective and efficient on its respective applications.

This work proposes an architecture for a decentralized movement control and a task allocation method based on Fuzzy Logic, that uses agents and environment information for inference. This model is a combination of strategies that has not been explored before. The proposal has been tested in a simulated scenario and compared against a centralized solution from our previous work in [3].

II. Background

A. Swarm Intelligence

Inspired by nature, *Swarm Intelligence* (SI) is a decentralized collective behavior for self-organized individuals [2]. Such individuals follow simple rules and own limited abilities to accomplish a complex task; but, through cooperation, a complex behavior emerges as a swarm by using crucial local mechanisms for the completion of the task [2] [19].

Models from nature are adopted for artificial swarm systems whose self-organization basis consist of: *positive and negative feedback*, *fluctuations in behavior*, and *multiple interactions* [1]. However, other features are taken into account such as *parallelism*, and *exploitation of direct (peer-to-peer) or indirect (via the environment) local communication mechanisms* [6]. According to Tan Ying *et al* [19], important characteristics as *parallelism*, *scalability*, *stability*, *economical*, and *energy efficiency* would allow systems to be built with great flexibility and robustness.

B. Fuzzy Logic

There is an important number of applications based on *Fuzzy Logic* theory, specially for control problems because of its features such as *robustness*, *ease to tweak*, *inclusion of several inputs and outputs*, *complexity of rules*, and *handle of non-linear systems difficult to model mathematically* [20].

The fuzzy logic approach for swarm movement from De Oliveira *et al* [9] is an example for controlling parameters like UAVs acceleration by using the *Takagi-Sugeno* fuzzy inference model, which is useful for controlling actions according to numerical outputs [21].

The work of De Oliveira *et al* inspired us to use the *Takagi-Sugeno* model as a solution for task allocation instead of controlling movement parameters. Task allocation problems have been solved with fuzzy logic before as in [14]. The fuzzy logic-based task allocator proposed here is part of a distributed swarm strategy as explained in section IV.

III. Methodology

In a previous work [3], a centralized solution for controlling a swarm of agents was developed, in which the communication occurs through a central control. This research is focused on a distributed, also called *decentralized* strategy based on a fuzzy logic approach for task allocation, in which the full logic resides on each agent for coordinating activities, by using a local communication module implemented with *one-way broadcast messages*; hence, all agents close enough to each other are able to receive the signals, but they do not answer to them, they just send their local relevant data.

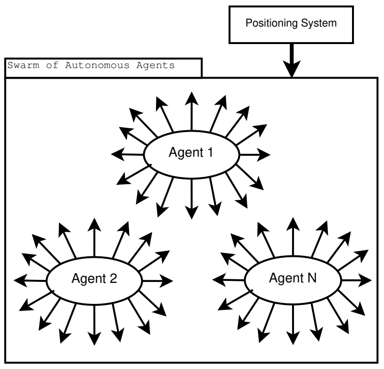
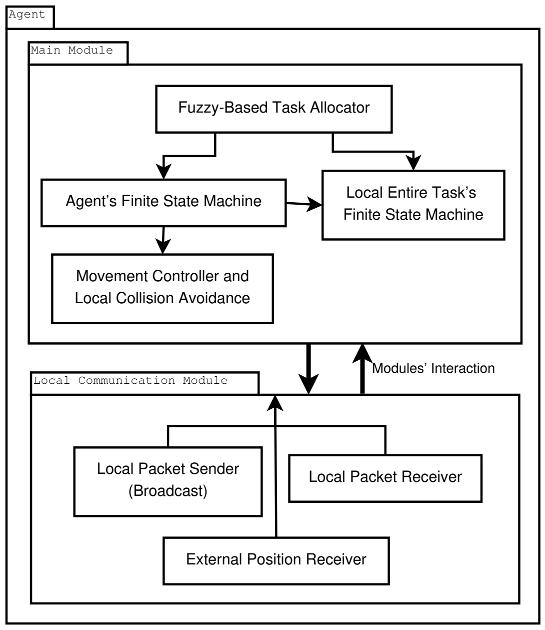


Fig. 1: Diagram for distributed swarm strategy


Agent
Main Module
Fuzzy-Based Task Allocator
Agent's Finite State Machine
Local Entire Task's Finite State Machine
Movement Controller and Local Collision Avoidance
Modules' Interaction
Local Communication Module
Local Packet Sender (Broadcast)
Local Packet Receiver
External Position Receiver
Fig. 2: Architecture for a decentralized agent

An agent-interaction diagram for the distributed solution is shown in Fig. 1. The arrows on each agent represent the message that is broadcasted through the environment; in consequence, near agents receive the message and act accordingly to their behavior. The swarm depends on a *positioning system* which provides the spatial coordinates.

Each agent supports an architecture that is composed by a *main* and a *local communication* module as depicted in Fig. 2. The main module is the central controller, which manages the agent's behavior by using a *finite state machine* (FSM), whose decisions allow the agent to be *moved* towards a task to be performed, or *avoid collisions* based on local data. The task is assigned by a *Fuzzy Logic-Based task allocator*, which uses data from the agent itself and its neighbors, as well as the local current state of the cooperative activity, which is controlled by another *FSM for the entire activity* to be solved. The local communication module receives the agent's position in the world frame through the *external position receiver*; also, broadcasts a message by using the *local packet sender* and receives data from the *local packet receiver*. The packets from the communication module are processed by the main module as part of the interaction among agents.

In order to test the effectiveness of the proposed strategy, the solution was implemented in a simulated environment using the Unity3D engine. The experiments performed were intended to be compared with the centralized solution developed in a previous work [3], thus the conditions for testing were the same, i.e., a swarm of drones that performed a virtual farming activity of sowing seeds in a field composed by a grid of 10 x 10 holes. Likewise, 30 experiments were performed using 1 to 30 agents in each trail; that is, up to 900 tests. Also, static obstacles were distributed on the field in the same positions as the past centralized strategy. The agents shared the same configuration and set of features. After each run, the time for accomplishing the task, the collision avoidances, as well as the counting of planted seeds, were taken for each agent. Comparisons between the centralized and decentralized strategy were established based on results to verify whether significant differences exist in accomplishing the task, the time and number of collision avoidance attempts; also, the effectiveness was tested considering whether all holes were planted.

IV. Distributed Swarm Method

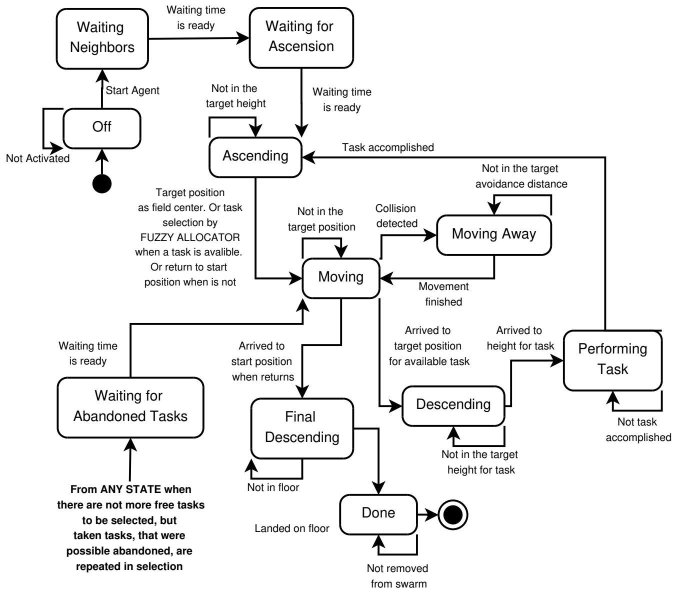
As in [3], the task defined is *sowing seeds*, the swarm of agents performs a set of movements for planting in a ground represented as Pg and modeled as a grid composed by r rows and c columns, where each element is a position in space defined by equation (1).

$$Pg = \{(x, y, z) : x, y, z \in \mathbb{R}, y = 0\} \tag{1}$$

Each point in Pg is separated by a distance sep from each other. Pg is represented as a one-dimensional array which resides on each agent, and it is updated with the local information gathered from its agent's neighbors, which allows to select the position where the agent needs to go according to its internal and external data, as well as the fuzzy strategy to choose a place to seed and collaborate mutually to finish the whole activity.

A. Decentralized Agent Behavior

The real-time behavior established by the FSM from our previous work was modified for a decentralized agent A, as described in Fig. 3. The agent starts with an initialization procedure, which is included as part of the strategy such as the ground configuration as described before, a data collection about neighbors, the positions of static obstacles, and starting the local communication module, which operates within a radius r_{lcom}. At the beginning, each agent calculates the center position $\vec{Pg_{center}}$ defined by equation (2) for the field Pg and changes its state to *Waiting Neighbors*, in which the agent remains inactive for a time t_{wn} while processing the local information received by neighbors.



Waiting Neighbors
Waiting time is ready
Waiting for Ascension
Start Agent
Not in the target height
Waiting time is ready
Off
Not Activated
Ascending
Task accomplished
Target position as field center. Or task selection by FUZZY ALLOCATOR when a task is avalible. Or return to start position when is not
Not in the target position
Collision detected
Not in the target avoidance distance
Moving Away
Moving
Movement finished
Waiting time is ready
Arrived to start position when returns
Arrived to target position for available task
Arrived to height for task
Performing Task
Waiting for Abandoned Tasks
Final Descending
Descending
Not task accomplished
Not in the target height for task
From ANY STATE when there are not more free tasks to be selected, but taken tasks, that were possible abandoned, are repeated in selection
Not in floor
Done
Landed on floor
Not removed from swarm

Fig. 3: Finite state machine (FSM) of a distributed agent

$$\vec{Pg_{center}} = \frac{\sum_{i=1}^{r \cdot c} \vec{Pg_i}}{r \cdot c} : \vec{Pg_i} \in \mathbb{R}^3, i, r, c \in \mathbb{Z}^+ \tag{2}$$

An agent A continually receives packets from neighbors, as well as it sends its own data as a broadcast message. The received data from a neighbor A_{ng} is composed by: its current position in space, a flag that denotes whether it is working or not, and the time since it was turn on. When the agent A receives a packet from a neighbor A_{ng}, A adds two neighbors to the data collection, A_{ng} and its neighbor A_{ng2}, but is not part of A yet. When A_{ng} sends a packet, A_{ng2} is not always the same on every dispatching, because A_{ng2} is shifted among the neighbors from the collection inside A_{ng}, this strategy allows the data to be spread out from agents that are not reached by others, note that A does not have to consider itself as neighbor in case that an A_{ng2} agent is A. Additionally, the *time since an agent was turn on* is used to add a more recent neighbor data, with the shortest time to A, in case that an existent agent is received.

The *current position in space* from neighbors is used to obtain the swarm center position $\vec{Ps_{center}}$, given by equation (3), which considers the current set of neighbors of size N_a with the agent A included. As a consequence of the continuous local communication, the position vector $\vec{Ps_{center}}$ changes constantly while neighbors are added (N_a increases) until t_{wn} ends in the *Waiting Neighbors* state, then the behavior is changed to *Waiting for Ascension* state. It is important to mention that the timer, which controls the finishing of t_{wn}, is reset every time a neighbor is added, hence other neighbors have the same chance to be part of agent A under the same period t_{wn}.

$$\vec{Ps_{center}} = \frac{\sum_{i=1}^{N_a} \vec{pa_i}}{N_a} : \vec{Ps_i} \in \mathbb{R}^3, i, N_a \in \mathbb{Z}^+ \tag{3}$$

The state *Waiting for Ascension* keeps the agent on the ground until a time t_{wa} is accomplished. This time t_{wa} allows other agents, that are already ascending, to move with a reduced possibility of collisions at the starting of the task, which saves time. Because of delays in local communications and a changing t_{wn} time, the resultant swarm from the local communication coverage is not going to ascend at the same time as in the centralized approach.

Considering *stability* and *consistency*, neighbors that were identified inside an agent A have a *lifetime* t_{lfn}, which is constantly checked in order to remove an old neighbor that might have been not working due to a failure or any other circumstances, such as an *out of range* case. The timer that controls the lifetime for a neighbor A_{ng} is also reset every time an agent A receives a packet from A_{ng}, as well as A_{ng2}, considering that broadcasting is persistent.

When the time t_{wa} is finished, the FSM changes from *Waiting for Ascension* to *Ascending* state to a defined height h_m. As in [3], the ascending and descending behaviors updates the agent position $\vec{pa}$ by considering a speed s_v for an upright displacement towards h_m (movement height) or h_p (planting height) for a period of time Δt, as defined by equation (4), which uses the unit vector $\hat{\mathbf{j}}$ to move vertically. This equation takes into account an ascending movement, and; for descending, the unit vector $\hat{\mathbf{j}}$ is inverted.

$$\vec{pa} = \vec{pa} + s_v \Delta t \hat{\mathbf{j}} : \vec{pa}, \hat{\mathbf{j}} \in \mathbb{R}^3, s_v \in \mathbb{R} \tag{4}$$

After the agent A reaches the height h_m, the target position to move is calculated considering the ground center $\vec{Pg_{center}}$ relative to the swarm center position $\vec{Ps_{center}}$, as defined in equation (5); then, its state changes to *Moving*, and A goes to $\vec{pags_{center}}$ throughout a straight line. Considering that $\vec{pags_{center}}$ is the current target position $\vec{pt}$ for A, then $\vec{pt} = \vec{pags_{center}}$; that is, agent A travels from its current position $\vec{pa}$ to a target position $\vec{pt}$ with speed s_m, and following a direction given by the unit vector defined in (6), which is used in the movement equation (7) that allows the agent to advance for a time Δt until it arrives at the target $\vec{pt}$.

$$\vec{pags_{center}} = (\vec{pa} - \vec{Ps_{center}}) + \vec{Pg_{center}} : \\ \vec{pa}, Ps_{center}, \vec{Pg_{center}} \in \mathbb{R}^3 \tag{5}$$

$$\hat{\mathbf{dir}} = \frac{\vec{pt} - \vec{pa}}{||\vec{pt} - \vec{pa}||} : \vec{pt}, \vec{pa} \in \mathbb{R}^3 \tag{6}$$

$$\vec{pa} = \vec{pa} + s_m \Delta t \hat{\mathbf{dir}} : \vec{pa}, \hat{\mathbf{dir}} \in \mathbb{R}^3, s_m \in \mathbb{R} \tag{7}$$

Due to a *tolerance time* t_{st}, which is calculated before entering to *Moving* state, A might not arrive exactly to $\vec{pags_{center}}$. This time t_{st} uses the swarm radius r_{sw} that is obtained by measuring the distance from the center of the swarm to the farthest agent inside, the distance between the swarm center and the ground center d_{sg}, and the moving speed s_m, as described in equation (8). While in *Moving* state, if t_{st} is reached, the agent stops to move towards the ground center and starts to sowing seeds according to the fuzzy logic-based allocator. This time tolerance is a mechanism to save time, that can be wasted due to delays produced by the collision avoidance interactions.

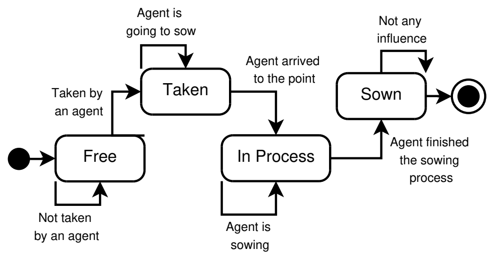


Fig. 4: Finite state machine for sowing a seed [3]

$$t_{st} = \frac{2r_{sw} + d_{sg}}{s_m} : r_{sw}, d_{sg}, s_m \in \mathbb{R}^+ \tag{8}$$

Collisions are managed locally by using positions from the collection of neighbors and static obstacles; thus, a warning collision radius r_c has to be smaller than the local communication coverage r_{lcom}, but large enough to prevent impacts. The collision avoidance algorithm designed in [3] was used, which forces an agent to change to a *Moving Away* state and travel a distance d_c away from the other agents, then return to *Moving* again after avoiding a collision. This algorithm is now distributed since it resides on board at each agent and its effectiveness depends on local communication delays.

Once the fuzzy logic strategy assigns a hole, the agent changes or is kept in *Moving* state towards its new target position $\vec{pt}$, avoiding collisions if needed, then arrives to its goal and changes to *Descending* state, descends to an specific height h_p and switches to *Performing Task* state in which a seed is released in the available hole in a time t_p, then ascends according to the *Ascending* state, and moves to the next available target after a new fuzzy logic-based calculation is done. This process is performed until all holes are filled, which is controlled locally to avoid task re-doing. The states for a hole are updated by the agents according to their actions. Fig. 4 shows these states in the FSM.

Agent A changes the state of a hole from *Free* to *Taken* when a free place is found; then, from *Taken* to *In Process* state when the agent is in the *Performing Task* state, and finally, *Sown* state is achieved when the agent dispatches the seed. Agents are not able to take other holes than the *free* ones, except if one of them is in *taken* state and it is considered *abandoned*, which might happen when there are not free places to seed on, and the fuzzy logic allocator is constantly selecting the same hole in *Taken* state. It should be noted that the whole activity is completed when all holes are ready (in *sown* state), and taken holes can still be selected because the most promising agents regarding a hole (free or taken) can sow it. In this case, agent A changes its current state to *Waiting for Abandoned Tasks*, as in Fig. 3, moving to a twice-agent-size distance, until a time t_{wat} is accomplished, and then it goes to the *Taken* place.

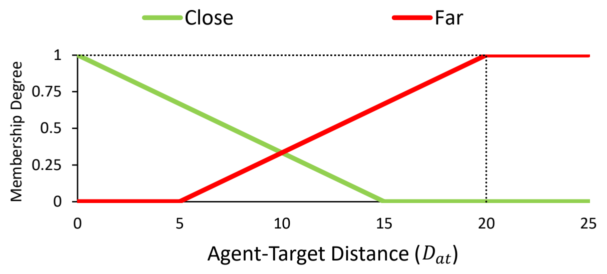


Fig. 5: Membership functions for the variable *Distance Agent-Target* ($\boldsymbol{D_{at}}$)

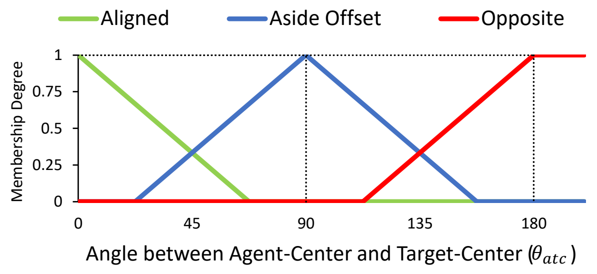


Fig. 6: Membership functions for the variable *Angle between* $\boldsymbol{Agent\text{-}\vec{Center}}$ *and* $\boldsymbol{Target\text{-}\vec{Center}}$ ($\boldsymbol{\theta_{atc}}$)

The taken holes can be own by different agents according to the internal data Pg in agent A. Additional parameters are associated with the local data collection of places (holes) such as: a flag f_{pa} that denotes whether a hole is processed by the agent A, and the identifier (ID) from the agent that is actually possessing, which can be any agent in the swarm.

Agent A transmits part of its collection of holes parameters to neighbors for keeping data updated; thus, neighbors and places are transmitted together in one packet through a broadcast message as mentioned previously. When a packet is received by agent A, the local data for holes in A is updated as long as f_{pa} for each hole is *false*, e.g., if a place is in *Taken* state, and this same place is in *Sown* state in the received data, then the information is replaced together with the ID of the agent that worked on that place; however, when there is a conflict with taken places, the data from the nearest agent to the place in dispute is chosen and forces agent A to select another hole, in case A is not the nearest one; and possibly trigger the rule of *abandoned* tasks.

Finally, the strategy to select places can be easily replaced, by either using the *immediate nearest place commission* as in [3], or the *fuzzy logic-based task allocation* explained below. Therefore, the model described in this section is flexible regarding a selection of the strategy.

B. Fuzzy Logic-based Task Allocator

In order to select a task to be processed from a set of *free* or *taken* places, a *Fuzzy Logic-based Task Allocation* algorithm was implemented by using input membership functions for two linguistic variables, which are:

- ***Distance Agent-Target*** ($\boldsymbol{D_{at}}$), which is the real-time distance given in meters from the agent to a possible target to be selected. Its membership functions are depicted in Fig. 5 and denote the fuzzy sets *Close* and *Far*.
- ***Angle between*** $\boldsymbol{Agent\text{-}\vec{Center}}$ ***and*** $\boldsymbol{Target\text{-}\vec{Center}}$ ($\boldsymbol{\theta_{atc}}$), that is a value given in degrees and obtained from position vectors $\vec{pa}$ (agent position), $\vec{Pg_i}$ (position for target i), and $\vec{Pg_{center}}$ (ground center position), thus the two displacement vectors, for calculating θ_{atc}, are defined in equations (9) and (10). Fig. 6 represents the membership functions for the fuzzy sets *Aligned*, *Aside Offset*, and *Opposite*.

TABLE I: AND rules for fuzzy logic inference with z output levels per pair

AND	**Aligned**	**Aside Offset**	**Opposite**
Close	1.00	0.75	0.25
Far	0.50	0.25	0.1

$$Agent\text{-}\vec{Center} = \vec{pa} - \vec{Pg_{center}} : \vec{pa}, \vec{Pg_{center}} \in \mathbb{R}^3 \qquad (9)$$

$$Target\text{-}\vec{Center} = \vec{Pg_i} - \vec{Pg_{center}} : \vec{Pg_i}, \vec{Pg_{center}} \in \mathbb{R}^3 \quad (10)$$

The crisp values that compose these functions are based on a simulated environment, where an agent is represented as a sphere of one meter of diameter and the ground is a field of 40x40 meters, thus variables D_{at} and θ_{atc} have to be adjusted to the conditions given by these parameters; in this case, the values for meters and degrees that delimits the starting and final points (0m, 5m, 15m, 20m; 0°, 22.5°, 67.5°, 90°, 112.5°,157.5°,180°) as well as the intersection points (10m; 45°, 135°) for the membership functions were chosen by experimentation in favor of better results by considering a linear shape.

The output obtained by the fuzzy inference process is based on the *Takagi-Sugeno* method as proposed in [21]. The solution is adjusted to a zero-order model; that is, the output membership function is a constant value z associated with each rule. The firing strength for a rule i is defined as w_i, which is obtained by the fuzzy AND operation given by the minimal value from the evaluation of membership functions values (0-1) when crisp values are taken as inputs for those functions. The final output from the fuzzy process is the weighted average of all N rules outputs, as shown in equation (11). We call the value of F_c as the *Factor of Choice* for a candidate target to be selected, thus the agent will iterate through all the targets (holes) in order to find the higher F_c, which will define the best place to go as result of this fuzzy inference process.

$$F_c = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i} : w_i, z_i \in \mathbb{R}, i, N \in \mathbb{Z}^+ \qquad (11)$$

The rules are all AND conditions as defined in Table I, in

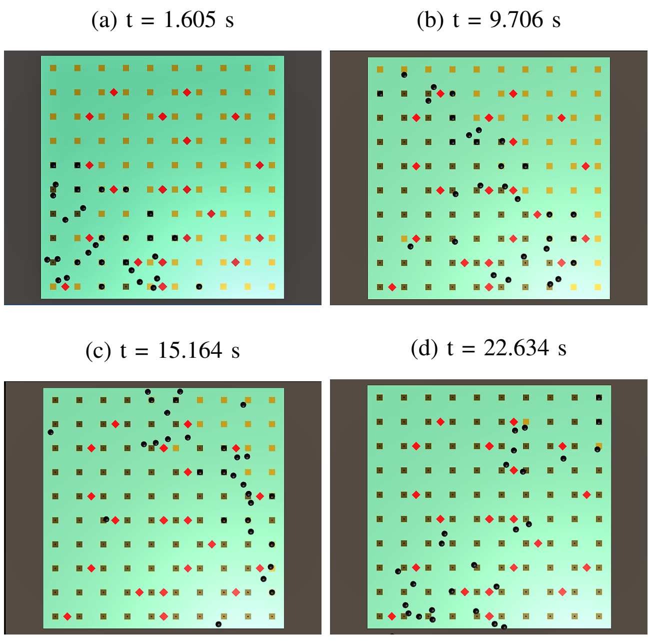


Fig. 7: Simulation screen-shots for different times considering a group of 30 drones: Centralized Approach

which the content of the cells are the z values selected for this strategy. Columns represent the fuzzy sets for variable θ_{atc} and rows denote the sets for variable D_{at}. The z were intuitively selected, therefore a more methodical criteria could be used for selecting these values.

V. Results

For testing purposes, a simulation was run for groups from 1 to 30 agents with 30 experiments for each group. The field for the experiments was configured as a grid of $10x10$ with 20 obstacles.

The component's programming paradigm used by the simulator *Unity3D* allows the decentralized agents to be implemented as objects with the same individual behavior, which uses two components, one for the FSM as presented in Fig. 3 which contains local information for the ground states as shown in Fig. 4, and the other for the communication module, that generates an expansive wave for broadcasting constantly, using the same radius coverage for all agents.

As shown in Fig. 7 and 8, the agents, denoted as circles, are spread out to solve the task on the field at different times for a swarm of 30 drones. The holes are represented by squares along the ground and the obstacles by diamonds. Note the differences between Fig. 7 (centralized approach) and Fig. 8 (distributed approach) considering the same points in time. The central controller uses a strategy of selecting the immediate nearest free place to be sowing, while the distributed control uses the fuzzy allocation strategy to select those places by trying initially to keep a formation towards the center of the field, and next, spread out from that sector.

The planting time for each swarm of agents was registered and compared among themselves to verify the improvement

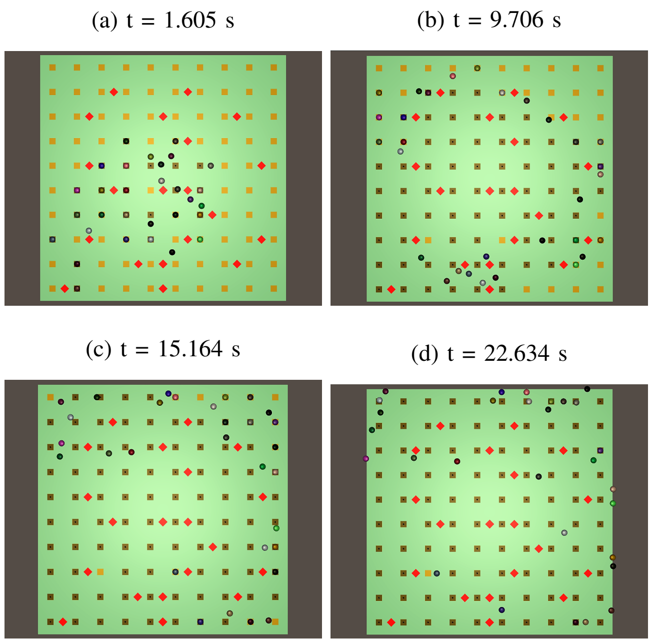


Fig. 8: Simulation screen-shots for different times considering a group of 30 drones: Distributed Approach

of time as the number of individuals increases. A relationship between the number of drones and the time spent to finishing the task was established as the function $y = \frac{a}{x+b}$ because the data behaves in a similar way to the centralized approach. A non-linear regression is described in Table II, with an estimation of goodness of 0.9757. Fig. 9 shows this behavior in which a solid red line represents the found model from the non-linear regression applied to our proposed distributed approach; also, a dotted blue line from the model established in our previous work for the centralized approach is drawn. Note a high similarity in both lines. Also, observe that the data owns considerable outliers in some number of drones, which was not presented in the centralized approach.

For this proposed strategy, the number of drones is compared against the number of collision avoidance, as seen in Fig. 10 the behavior is linear. A simple linear regression with an estimation of goodness of 0.3756 was calculated to establish this relationship. A regression equation described in Table III was found $(F(1, 898) = 147.5, p < 2.2x10^{-16})$, with an R^2 of 0.1411. The estimation of goodness is a low value and the relationship can be considered weak; despite of important outliers which affects this calculation as Fig. 10 shows, it was possible to find a model according to the data concentration; a solid red line that represents the model for the distributed approach is depicted as well as a dashed blue line that denotes the model obtained in our previous work. Note that, with few group of agents, the number of collision avoidances for the distributed approach is greater than the centralized one; however, as the number of agents increases, the collisions attempts are reduced.

Another comparison between the centralized and the dis-

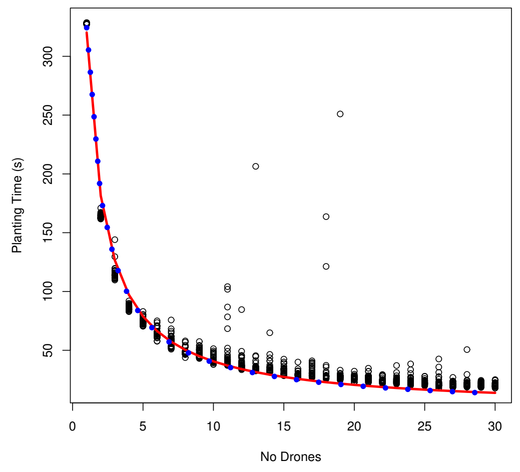


Fig. 9: Curve: *Number of Drones* vs *Planting Time* for distributed strategy: Solid line represents distributed approach and dotted line is the centralized approach

TABLE II: Non-linear regression results for time in the distributed approach

Parameter	Estimate	Std. Error	t value	Pr($>|t|$)
a	418.38302	5.72731	73.05	<2e-16
b	0.30636	0.02284	13.41	<2e-16

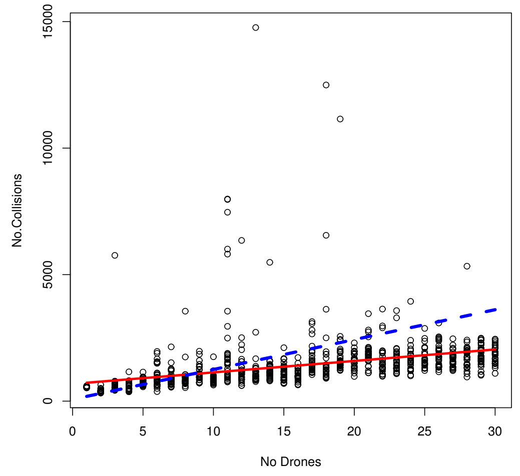


Fig. 10: Curve: *Number of Drones* vs *Number of Collision Avoidances* for distributed strategy: Solid line represents distributed approach and dashed line is the centralized approach

TABLE III: Linear regression results for collision avoidances in the distributed approach

Parameter	Estimate	Std. Error	t value	Pr($>|t|$)
a	682.551	66.292	10.30	<2e-16
b	45.357	3.734	12.15	<2e-16

TABLE IV: Group of agents with p-value $< .05$ for a *t-test* considering *Planting Times*. Levene's test p-value is showed for each group

No. Drones	μt_1	μt_2	$\mu t_1 - \mu t_2$	Levene's test P-value
1	330.905	327.939	2.966	$<$ **.05**
4	87.548	86.145	1.403	$>$ .05
8	52.098	49.061	3.037	$>$ .05
20	28.923	26.539	2.384	$>$ .05
21	28.256	26.795	1.461	$<$ **.05**
22	27.804	24.436	3.368	$>$ .05
23	27.749	24.532	3.217	$>$ .05
24	26.824	24.306	2.518	$<$ **.05**
25	26.834	22.639	4.195	$>$ .05
26	27.119	22.890	4.229	$<$ **.05**
27	26.086	21.218	4.868	$>$ .05
28	25.707	22.343	3.364	$>$ .05
29	25.231	20.727	4.504	$>$ .05
30	24.425	20.579	3.846	$>$ .05

TABLE V: Group of agents with p-value $< .05$ for a *t-test* considering *Collision Avoidances*. Levene's test p-value is showed for each group

No. Drones	μc_1	μc_2	$\mu c_1 - \mu c_2$	Levene's test P-value
2	456.533	401.733	54.800	$>$ .05
4	647.833	571.667	76.166	$>$ .05
10	1106.000	917.066	188.934	$>$ .05
14	1580.767	1302.933	277.774	$>$ .05
15	1779.667	1151.333	628.334	$<$ **.05**
16	1997.733	1059.233	938.500	$>$ .05
17	2005.633	1662.000	343.633	$>$ .05
20	2507.067	1640.100	866.967	$<$ **.05**
21	2665.100	1881.567	783.533	$>$ .05
22	2711.233	1619.800	1091.433	$>$ .05
23	2878.300	1648.833	1229.467	$>$ .05
24	3011.933	1641.467	1370.466	$>$ .05
25	3099.767	1702.800	1396.967	$>$ .05
26	3108.967	1839.100	1269.867	$<$ **.05**
27	3222.867	1696.933	1525.934	$>$ .05
28	3300.767	1831.233	1469.534	$>$ .05
29	3584.700	1819.833	1764.867	$<$ **.05**
30	3639.500	1842.467	1797.033	$>$ .05

tributed approach is that both data, for each group of agents, were processed under the assumption of a normal distribution given by the *central limit theorem* because of the size of the data (≥ 30); thus, a *t-test*, for determining a significant difference in *planting time means* (μt) and *collision avoidances means* (μc) for both strategies, was applied and calculated properly, according with the results obtained from a *Levene's test* for homogeneity of variances. The alternative hypothesis considers that, the centralized group's times or collision avoidance, are greater than the distributed group's times or collision avoidance ($\mu t_1 > \mu t_2$; $\mu c_1 > \mu c_2$). Tables IV and V show the results for p-values according to the previous description, only those groups of agents whose null hypothesis is rejected in favor of the alternative hypothesis for the *t-test* are taken into account. Improvements for high number of agents can be noted in the distributed fuzzy logic-based approach; the reduction of collision avoidance is remarkable.

In terms of *scalability*, as well as in the centralized approach, the system completed every task with nothing more than adding or removing agents without any modification on the configuration or the implementation; however, in some experiments for large-sized swarms, there were between 1 to 6 extra seeds released, but *effectiveness* was not affected. Also, considering *stability*, in some experiments agents were removed in real-time and the completion of the activity was not affected.

VI. Conclusions

This paper presents a distributed strategy for controlling a swarm of agents and tasks assignment based on a fuzzy logic approach. The results compare two parameters; *completion times* and *collision attempts*, between a distributed and a centralized strategy, under the same conditions in a simulation.

The time for performing the task decreases as the number of agents is increased by following a non-linear behavior with a clearly similarity between strategies; there were groups of individuals with an improvement considering the distributed method, specially for groups of 20 or more agents; hence, this new strategy is able to reduce times for large groups of agents compare to our previous solution.

Regarding collisions, there is a contrast between both solutions in which the distributed strategy registers more possible collisions for small groups; but, for larger groups, the reduction is considerable, near 50% of improvement for the largest swarm; thus, the distributed solution avoids less collisions than the centralized approach for large groups of agents.

For implementing this strategy in real environments the communication delays have to be considered, since these could increase the collisions attempts, and tasks can be unnecessarily dispatched.

Finally, these findings can be useful for *energy saving* considering that real UAVs consume high amounts of energy that need to be store in small suitable batteries. Also, a full decentralized swarm coordination algorithm; like the one proposed here, meets the requirements for achieving the completion of an activity in terms of scalability and stability, properties from an artificial swarm of individuals.

For future work, the strategy is intended to be refined to reduce outliers and wasted resources (extra seeds), and implemented with real UAVs by considering the addition of sensors to get the autonomy required for an out-door environment.

Acknowledgement

This research was developed using the infrastructure provided by the ANDAMIOS Project supported by the Secretaría de Educación Superior, Ciencia, Tecnología e Innovación - SENESCYT of Ecuador.

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*, 1999.

[2] Y. Zhang, P. Agarwal, V. Bhatnagar, S. Balochian, and J. Yan, "Swarm Intelligence and Its Applications," *Hindawi Publishing Corporation: The ScientificWorld Journal*, vol. 2013, p. 3, 2013.

[3] K. Loayza, P. Lucas, and E. Pelaez, "A centralized control of movements using a collision avoidance algorithm for a swarm of autonomous agents," in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*. Salinas: IEEE, oct 2017, pp. 1–6.

[4] S. Mostaghim, C. Steup, and F. Witt, "Energy Aware Particle Swarm Optimization as search mechanism for aerial micro-robots," in *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*, 2017.

[5] V. D. Koshur, "Reinforcement swarm intelligence in the global optimization method via neuro-fuzzy control of the search process," *Optical Memory and Neural Networks*, vol. 24, no. 2, pp. 102–108, 2015.

[6] Y. Meng, O. Kazeem, and J. C. Muller, "A Swarm Intelligence Based Coordination Algorithm for Distributed Multi-Agent Systems," *Integration of Knowledge Intensive Multi-Agent Systems, 2007. KIMAS 2007. International Conference on*, pp. 294–299, 2007.

[7] D. H. Kim, H. Wang, and S. Shin, "Decentralized Control of Autonomous Swarm Systems Using Artificial Potential Functions: Analytical Design Guidelines," *Journal of Intelligent and Robotic Systems*, vol. 45, no. 4, pp. 369–394, 2006.

[8] R. Wang, X. Dong, Q. Li, and Z. Ren, "Distributed Adaptive Formation Control for Linear Swarm Systems with Time-Varying Formation and Switching Topologies," *IEEE Access*, vol. 4, pp. 8995–9004, 2016.

[9] L. F. De Oliveira, F. B. De Lima, S. C. Oliveira, and C. J. Bastos-Filho, "A fuzzy-swarm based approach for the coordination of unmanned aerial vehicles," *Journal of Intelligent and Fuzzy Systems*, vol. 31, no. 3, pp. 1513–1520, 2016.

[10] Z. Minchev, O. Manolov, S. Noykov, U. Witkowski, and U. Riickert, "Fuzzy logic based intelligent motion control of robot swarm simulated by Khepera robots," *2004 2nd International IEEE Conference on 'Intelligent Systems'. Proceedings (IEEE Cat. No.04EX791)*, vol. 1, 2004.

[11] B. J. O. De Souza and M. Endler, "Coordinating movement within swarms of UAVs through mobile networks," *2015 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2015*, no. April, pp. 154–159, 2015.

[12] Y. Jin, Y. Wu, and N. Fan, "Research on distributed cooperative control of swarm UAVs for persistent coverage," *Proceedings of the 33rd Chinese Control Conference, CCC 2014*, pp. 1162–1167, 2014.

[13] G. Lionis and K. J. Kyriakopoulos, "Decentralized lattice formation control for micro robotic swarms," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 3756–3761, 2009.

[14] U. Kaushal and A. Kumar, "Fuzzy Logic-Based Task Allocation Model with Memory Constraint for Distributed Real-Time System," *The IUP Journal of Information Technology*, vol. X, no. 4, pp. 34–51, 2014.

[15] S. H. Ling, F. Jiang, H. T. Nguyen, and K. Y. Chan, "Hybrid Fuzzy Logic-Based Particle Swarm Optimization for Flow Shop Scheduling Problem," *International Journal of Computational Intelligence and Applications*, vol. 10, no. 03, pp. 335–356, 2011.

[16] T. Long, X. Huo, Y. Niu, and L. Shen, "Distributed cooperative planning for multiple UAVs based on agent negotiation," *2006 International Conference on Computational Intelligence and Security (IEEE Cat. No.06EX1512)*, pp. 4 pp.—CD—-ROM, 2006.

[17] N. Basilico and F. Amigoni, "Exploration strategies based on multicriteria decision making for searching environments in rescue operations," *Autonomous Robots*, vol. 31, no. 4, pp. 401–417, 2011.

[18] S. Kraus and T. Plotkin, "Algorithms of distributed task allocation for cooperative agents," *Theoretical Computer Science*, vol. 242, no. 1-2, pp. 1–27, 2000.

[19] Y. Tan and Z.-y. Zheng, "Research Advance in Swarm Robotics," *Defence Technology*, vol. 9, no. 1, pp. 18–39, 2013.

[20] R. Lasri, I. Rojas, H. Pomares, and Z. Sadouq, "Explaining how intelligent control has improved the way we live: A survey on the use of Fuzzy Logic Controllers in daily human life," in *2011 International Conference on Multimedia Computing and Systems*. IEEE, apr 2011, pp. 1–6.

[21] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, 1985.