Swarm Collective Wisdom: A Fuzzy-Based Consensus Approach for Evaluating Agents Confidence in Global States

Aya Hussein, Sondoss Elsawah, and Hussein A. Abbass
School of Engineering and Information Technology
University of New South Wales
Canberra, Australia
a.hussein@student.adfa.edu.au, s.elsawah@adfa.edu.au, h.abbass@unsw.edu.au

***Abstract*—Consensus achievement is a class of problems in which a group of agents, such as a swarm, needs to collectively reach a common decision to select one of the available options. Many consensus achievement strategies were proposed in which an agent forms its opinion and exchanges it with the other agents to reach a collective decision. To facilitate the decision making process, agents which are highly confident in their opinions are commonly given a higher chance to influence the collective's decision making. However, the use of subjective metrics for confidence could degrade the performance of the state-of-the-art algorithms in complex scenarios where agents with wrong opinions can be the most confident. To tackle this problem, we propose an objective metric for confidence by using experience to learn the mapping between the information available to an agent and the probability that the agent's opinion is correct. To compute its confidence level, an agent feeds data from its local observations, as well as the received neighbours' opinions, into a fuzzy inference system (FIS) that uses these inputs to estimate confidence. The proposed strategy is distributed and it requires the agents to communicate locally using messages containing only their ID and opinions. Our strategy is evaluated under scenarios with different levels of complexity. The results show that our algorithm outperforms the state-of-the-art algorithms in terms of its accuracy, task time, and ability to reach majority. The proposed approach was also shown to maintain its success, even in the most complex environments.**

***Index Terms*—Swarm Decision Making, Best-of-n Problem, Fuzzy Inference System, Collective Decision Making, Self-Organisation**

I. Introduction

Collective decision making can be defined as "the phenomenon whereby a collective of agents makes a choice in a way that, once made, it is no longer attributable to any of the individual agents" ([1, p.1]). In the field of swarm and multi-agent systems, Brambilla et al. [2] classified collective decision making problems into two classes: consensus achievement and task allocation. Consensus achievement scenarios are those in which the swarm members need to collectively reach a common decision to select one of the available options. On the other hand, task allocation problems require agents to assign tasks to themselves to maximise the overall mission performance. This paper is focused on consensus achievement problems. Consensus achievement has received the interest of many research studies on swarm and multi-agent systems (e.g. [1], [3]–[9]) due to its widespread applications. Examples of these applications in the literature include: robot selection within human interaction with multi-robots [10], the selection of the shortest path to be traversed by the swarm [11], swarm leader election [12], best site selection [6], and abnormal behaviour detection [13].

Several studies (e.g. [3], [5], [14], [15]) proposed consensus achievement strategies using opinion-based approaches. In opinion-based approaches, agents have an explicit representation of their opinion but the specifics of how opinions are exchanged and how decisions are made can vary between different strategies [1]. To facilitate the decision making process, some strategies give agents that are highly confident in their opinions a higher chance to influence the collective decision. For instance, in [3]–[6] an agent subjectively calculates the level of confidence in its opinion using only its own observations. We describe this way of confidence calculation as subjective because it relies only on an agent's own observations regardless of the actual probability of the opinion to be correct. Agents then broadcast their opinions for a duration proportional to their confidence level such that the opinions of highly confident agents can be received by many agents. Due to the subjectivity in confidence estimation, it is not hard to imagine situations in which agents with the wrong opinion having high levels of confidence. For example, agents that encounter the same set of observations again and again will be more confident than those encountering a diverse set observations, possibly with conflicting cues. Facilitating the propagation of wrong opinions strongly disturbs the decision making process as it can result in an incorrect collective decision, lack of agreement on the final decision, or at least inefficient decision making.

This work aims to propose a consensus achievement algorithm that avoids the weaknesses of the existing algorithms by designing an objective confidence metric that can be calculated by each agent in a distributed way during the consensus achievement task. This facilitates the decision

This work was funded by the Australian Research Council Discovery Grant number DP160102037 and UNSW-Canberra.

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

making process as agents with wrong opinions are given lower chances to broadcast their opinions which improves the swarm performance under different levels of complexity. Using the proposed metric, an agent calculates its confidence level using a pre-trained FIS model given data from both the agent's local observations and the opinions received from the other agents.

II. Related Work

Consensus achievement algorithms have been used in different swarm-based applications. In a consensus achievement scenario, a collective decision needs to be reached based on different pieces of evidence collected by the swarm members. Swarm members are typically of limited sensing and computational capabilities. However, by properly fusing their local observations, complex decision making problems can be solved. In this section, we present some examples of how consensus achievement algorithms were used to solve real-life problems. We follow this with a discussion on how the existing generic algorithms work and what aspects of these algorithms can be improved.

Several studies used consensus achievement algorithms to facilitate human-swarm interaction. For instance, Giusti et al. [14] used a swarm distributed consensus algorithm to enable a human to interact with a swarm using gestures. The human uses a predefined set of hand gestures to send commands to the swarm. Being spatially distributed, the robots capture images of the gesture from different viewpoints. Each robot performs some processing on its image to form an opinion on the gesture. Robots then exchange their opinions to reach a consensus so that the whole swarm can execute the command. Consensus achievement was also used to enable other intuitive human-to-swarm interfaces. In [10], Couture-Beil et al. used distributed face detection so that the robots can decide which individual robot the human is looking at to select for performing a task. Similarly, Nagi et al. [15] proposed an algorithm that allows a human to use a set of spatial gestures to select an individual or a subgroup of robots to command. In addition to human-swarm interfaces, the literature contains different applications of consensus achievement in swarm-based tasks including: shortest path selection [11], swarm leader election [12], best site selection [6], distributed feature detection [5], and abnormal behaviour detection [13]

Although many similarities exist between different consensus achievement problems, most of the existing algorithms are domain specific as they rely on exploiting particular features of the environment [6]. This limits the ability to deploy a consensus achievement algorithm designed to solve one problem in other problems. Recently, Valentini et al. [5] formulated a consensus achievement problem to serve as a benchmark problem for developing domain-agnostic collective decision making strategies. In this formulation, agents explore a grid-based environment to evaluate the abundance of an environmental feature that is scattered across the environment. The swarm is required to collectively determine whether or not the feature is frequent in the environment, i.e the feature exists in at least half of the cells. The problem resembles the swarm searching for precious metals, pollutants, or cancer cells. However, the feature is abstracted as the color of the cell (which can be black or white) to allow for designing general strategies that do not assume domain specific properties of the feature.

The complexity of the problem formulation, presented by [5], is influenced by the ratio of cells containing the feature in question [5] as well as the homogeneity of feature distribution [3]. As the ratio of white cells approaches 0.5, the problem becomes harder as it requires an agent to make a high number of observations before reaching a decision on which color is the most frequent. On the other hand, when this ratio approaches 1 in mostly white environments or 0 in mostly black environments, the agent needs to sample a lower number of cells before reaching a decision. So, the problem becomes easier in such environments.

The spatial distribution of the black and white cells in the environment is a another important factor that has a great impact on the complexity of the scenario [3]. A homogeneous environment with feature ratio r is an environment in which the color of a cell can be white with a probability r independently of other cells. Meanwhile, a non-homogeneous environment with feature ratio r has a continuous region of white cells constituting r of the environment area. Non-homogeneous environments increase the problem complexity as an agent's estimate will be biased by the regions it explores. This means that agents exploring different regions are expected to have widely different views of the environment.

Three state-of-the-art algorithms were proposed by previous studies to solve this problem: Voter Modulation [4], Direct Comparison (DC) [5], and Majority Modulation [3], [6]. While these algorithms have many similarities, they are mainly different with regard to what information are exchanged between agents and how agents update their opinions based on the received information. Starting with their similarities, in these algorithms agents can be in one of two phases: exploration and opinion dissemination. While in the exploration phase, an agent navigates through the environment to sense its distributed feature. At the end of the first exploration phase, each agent sets its opinion based on the most frequently encountered colour. The agent calculates its confidence as the ratio of the cells with the color associated with its opinion to the total number of cells encountered. That is, the confidence reaches its maximum level of 1 when all the cells encountered by the agent have the same color. In contrast, the lowest confidence level of 0.5 occurs when the agent observes the same number of black and white cells. In effect, in non-homogeneous environments, agents that explore only a single-coloured region will be the most confident regardless of the relative area of the region to the whole environment. On the contrary, agents that navigate through different regions of the environment will be less confident though their observations are more representative of the actual feature distribution.

After its exploration, an agent disseminates its opinion for a duration proportional to its confidence level. That is, agents that are highly confident are given more power to influence

the decision. In the Voter and Majority algorithms, agents exchange messages containing only their ID and opinion. The DC algorithm, however, requires the agents to send their confidence level together with their ID and opinion. When the dissemination phase ends, the agent starts another round of exploration-dissemination. The three existing algorithms differ in how agents fuse the received opinions to form their own opinion. In the Voter algorithm, an agent simply copies the opinion of a random neighbour. In the Majority algorithm, the agent applies a majority rule on all the received opinions including its own opinion and copies the opinion of the majority. In the DC, however, an agent copies the opinion of another agent if and only if the confidence of the received opinion is higher than its own confidence.

Valentini et al. [5] compared the three algorithms under different complexity levels by manipulating the feature ratio r. They found that, in easy settings (high feature ratio), DC had the best performance in terms of accuracy and speed. However the speed of the DC was the most sensitive to feature ratio. In complex settings, the algorithms showed a speed-accuracy trade-off; such that the Majority was the fastest but the least accurate. The Voter had a high accuracy similar to DC but it recorded the lowest speed.

III. Problem Formulation

In this work, we use the consensus achievement problem proposed in [5] and later used in [3], [9]. This abstract formulation allows for focusing on developing the decision-making algorithm itself rather than exploiting domain specifics to facilitate solving the problem. Thus, generic algorithms can be designed to be ported across different domains.

The problem can be formulated as follows: Consider an $L_{env} \times L_{env}$ grid-based environment, E, where each cell c_i is charachterised by a value of a binary feature f, such that $f : c_i \rightarrow \{0, 1\}, \forall c_i \in E$. The feature f represents the colour of the cell and can be either black or white. A swarm of N agents $\Pi = \{\pi_1, \pi_2, ...\pi_N\}$ is deployed in the environment which is bounded by four walls that are detectable by the swarm members. In the beginning of the mission, the swarm members are placed in an $L_{nest} \times L_{nest}$ nest in the top left corner of the environment with random positions and orientations. The swarm is required to explore the environment and perform collective decision making to decide which feature value is the most frequent in the environment.

The swarm members have some limitations on their actions, similar to the limitations described in [3]. First, an agent π_i can sense the colour of a cell c_j only if the position of π_i lies within the boundaries of c_j. In addition, at any time step an agent π_i can either sense the colour of a cell or disseminate its opinion, but not both. Meanwhile, π_i can listen to other agents at each time step, regardless of whether it is sensing or disseminating. Besides, the swarm members have no means for calculating their absolute positions within the environment. Thus, they can not determine which cells have or have not been visited by the swarm as a whole. An agent π_i can locally communicate with another agent π_j, if the distance between π_i and π_j is less than the communication range, R_{comm}. Using these local communications, estimates calculated by individual swarm members can be exchanged to form an overall estimate of the feature in question.

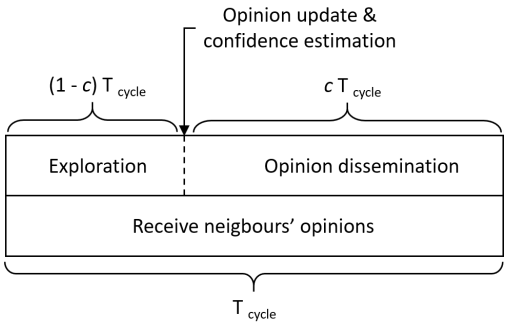

Fig. 1. The opinion update cycle. Confidence level is denoted by c.

IV. The Proposed Algorithm

Our proposed algorithm builds on the algorithms presented in [3]–[6] with the main difference being in confidence estimation. The details of the proposed algorithm are as follows:

A. Motion

An agent π_i navigates through the environment by performing a random walk which is a simple strategy that does not require agents to calculate their positions within the environment. Random walk is achieved by a straight line movement followed by an on-the-spot rotation. An agent π_i moves in a straight line for a duration t_s which is a random variable with an exponential distribution $t_s \sim EXP(\lambda)$. Then, π_i performs an on the-spot rotation with an angle drawn from a uniform distribution $U[\frac{-\pi}{2}, \frac{\pi}{2}]$. This sequence of straight line movements and rotations is repeated till the end of the simulation. Collision is resolved by a random rotation followed by a new cycle of random walk.

B. Opinion Update Cycle

The opinion update cycle consists of two sequential phases: exploration and opinion dissemination. While the length of an opinion update cycle is fixed to T_{cycle} time steps, the duration of each phase is dynamic and is calculated based on an agent's level of confidence, as shown in figure 1.

Exploration and opinion dissemination

Agent π_i keeps a record of the numbers of black n_{black} and white n_{white} cells it encounters within the environment. These numbers are set to zero in the beginning of the simulation and are then updated after the colour of each cell is sensed. At the end of each exploration phase, π_i calculates its estimate e_i based on its observations, such that e_i is the ratio of white cells to the total number of cells observed. This estimate is then used as the agent's opinion Ω_i.

$$e_i = \{\frac{n_{white_i}}{n_{black_i} + n_{white_i}}\} \quad (1)$$

When an agent π_i moves to the opinion dissemination phase, it keeps disseminating its opinion Ω_i in each time step till the end of the belief update cycle. As stated previously, the duration of exploration and the opinion dissemination phases are determined based on the agent's level of confidence. However, in the first cycle, agents do not have the sufficient data to calculate their confidence, so the confidence level is set to its mid value of 0.5. The details of estimating the level of confidence in the subsequent cycles are described in IV-C.

Receiving neighbours' opinions

As mentioned earlier, in each time step, an agent can receive opinions transmitted by other agents within its own communication radius, R_{comm}. Similar to the algorithm in [3], each agent keeps track of all the received opinions by integrating them into one local variable: opinion concentration, γ. In effect, γ can be thought of as a spatio-temporal integration of the other agents' opinions as perceived by an agent. In the beginning of the simulation, γ is set to 0.5, which then gets updated during the course of the simulation as follows:

- Agent π_i keeps a list of all opinions received in the current belief update cycle. In the beginning of each cycle, the list is cleared.
- When agent π_i receives the opinion Ω_j of an agent π_j, π_i searches its list to check whether it contains an entry for π_j. If an entry is found for π_j, then Ω_j is ignored. Otherwise, Ω_j is added to the list and is used to update γ, according to the equation:

$$\gamma_i = w\gamma_i + (1 - w)\Omega_j \tag{2}$$

The weight w determines the influence of a newly received opinion Ω_j on γ_i. We set w to 0.9 to limit this influence so that γ_i becomes a robust representation of all the received opinions. When γ_i approaches 0 (or 1), it reflects the fact that most opinions received by the agent were 0 (or 1).

Opinion update and decision making

At the end of each exploration phase, an agent π_i updates its opinion, by setting it to match its estimate e_i, before disseminating it. To facilitate reaching consensus, if an agent π_i perceives that the received opinions are sufficiently decisive (i.e. most agents have the same opinion), it makes a non revertible decision that is consistent with the received opinions. To achieve this, in each time step π_i compares γ_i against a threshold θ such that if $\gamma_i < \theta$ or $1 - \gamma_i < \theta$, π_i makes an irreversible decision and sets its opinion Ω_i to

$$\Omega_i = Round\{\gamma_i\} \tag{3}$$

C. Confidence Assessment

To objectively assess its level of confidence, an agent needs to estimate the expected accuracy of its opinion. Estimating the confidence based only on its direct observations can be highly misleading. For instance, encountering the same observation again and again will result in increasing the confidence, although this may result from a scenario where an agent is stuck in a small region. Thus, the level of confidence should not be evaluated based only on the direct observations. Although an agent has no means to access the ground truth to estimate its confidence during the task, it can learn from previous experiences how to map its state to an expected accuracy. That is, the task now is to define how to describe a state and to teach the agents how to estimate their level of confidence from their state. Below is a discussion on how our algorithm calculates the confidence level by describing the relevant state variables, the inference model used to map a state to a confidence level, and how machine learning is used to train this model.

Agent state characterisation

We propose characterising the state of an agent at a given point of time using the following four variables:

- The number of observations an agent made: we know from Statistics that the higher the number of observations sampled by an agent, the more reliable its estimation is. Thus, the level of confidence should be positively related to the number of observations. We acknowledge that this assumption may break down in adversarial environments, where the experience an agent gets exposed to may get manipulated by a second agent to deceive the former.
- The strength of the feature estimated by an agent: this variable has been used by previous algorithms [3]–[6] to determine agents' level of confidence. However, using it alone can be misleading as it does not capture the dependency between adjacent environment cells in non-homogeneous environments. This variable is calculated by an agent π_i as $|e_i - 0.5|$.
- Opinion concentration γ_i: perceiving the collective opinion of the other agents and the level of agreement among their opinions is crucial. That is, a collective opinion with a majority vote of 51% tells a different story than the same collective opinion but with a majority vote of 90%.
- The agreement between the agent's estimate e_i and the perceived collective opinion γ_i: when the agent's estimate leads to the same conclusion as the aggregated opinions of the other agents, the agent's confidence should be higher than when they lead to different conclusions.

The proposed state variables represent different but complementary sources of information that can be used to infer the expected accuracy of an agents' opinion. For instance, it is rational for an agent to assign a high level of confidence to its opinion when this opinion is based on a large number of observations, the observations are highly consistent (i.e, most observed cells are white or black), opinions received from other agents are highly consistent (most agents have the same opinion), and the agent's own estimate agrees with the aggregated opinions received from the other agents. On the contrary, if the agent has a large number of consistent observations but its estimate disagrees with the aggregated opinions of the other agents, it might be rational to lower its level of confidence as this state may stem from a scenario where all the agent's observations come from a single-colored region in the environment.

To estimate their confidence level, each agent determines its state using these four variables then uses some inference model to map its state to a confidence level. In fact, we have some ideas about how confidence levels should be assigned to some of the easily explainable states. Nevertheless, it is impractical to hard-code the mapping between all the states and the corresponding confidence levels for two main reasons. First, the state space, characterised by the previously mentioned four variables, is continuous. Even if the continuous variables are discretised, the resulting space will still be huge. Second, while we perceive states in terms of qualities like "large number of observations" , "highly consistent", and "highly confident", it is not straightforward to translate these qualities into numbers. That is, we may tell that an agent that made 500 observations should be *highly confident* in its opinion. However, we may not be able to accurately translate *highly confident* to a number in the interval between 80% and 100% on the confidence scale. Therefore, we use machine learning to learn these rules from previous experience. The details of the inference model and its rules are supplied in the next subsections.

Mapping states to confidence levels using FIS

We propose the use of a Fuzzy Inference System (FIS) to approximate the mapping between the states and their corresponding confidence levels. FIS is based on the fuzzy set theory, proposed by Zadeh in 1965 [16]. Fuzzy sets facilitate the use of linguistic variables to express rules and apply logic-like fuzzy operators on these rules to make inferences. We chose FIS due to its simplicity and low computational cost during the reasoning process. This makes it suitable for deployment on swarm agents that can be of limited computational capabilities. FIS works in a similar way to human reasoning which facilitates interpreting model results and analysing its performance.

The operation of FIS can be decomposed into three main steps, as follows:

1) Fuzzification: in this step, numerical input variables are fuzzified by calculating their degree of membership to the fuzzy sets. That is, the four numerical state variables are converted into fuzzy variables. We used five fuzzy sets for each of the first three variables, and only two sets for the fourth variable as it is binary.
2) Fuzzy inference: FIS has a database of IF-THEN rules that map fuzzy inputs to fuzzy outputs. Fuzzy variables calculated from the previous stage are checked against each of these rules to determine which rules are fired (or activated). Then the outputs from the fired rules are combined using fuzzy operators.
3) Defuzzification: fuzzy outputs generated from the previous step are converted back into crisp outputs so that they can be used by the agents as a numeric representation of their confidence level.

The database containing the fuzzy rules is a key element in any FIS. Generally, these rules can be supplied directly by a domain expert. For instance, in our case, we can define fuzzy rules like:

IF number of observations is very large **AND**
estimated feature strength is very high **AND**
opinion concentration is very high **AND**
local estimate agrees with opinion concentration
THEN confidence is very high.

Although such linguistic rules are intuitive, we prefer learning the rules from a set of training data rather than defining them ourselves. In fact, our database can have a maximum of $5 \times 5 \times 5 \times 2 = 250$ rules and defining such a large number of rules can be tedious. Besides, using the training data to learn the rules can help increase their precision.

Learning the fuzzy rules

To obtain a set of training data, we ran the algorithm under 6 different complexity levels: 3 feature ratios (r = 0.75, r = 0.65, and r = 0.55) $\times$ 2 homogeneity levels (homogeneous and non-homogeneous). Each complexity level was run 10 iterations. That is, a total of 60 simulation runs were used to generate the training data. In all these runs, the confidence level for all the agents is set to 0.5 and kept fixed throughout the simulation. We logged the state of an agent and whether its opinion was correct in the end of each belief update cycle. This resulted in about 38K rows of data. Then, we quantised the state variables, grouped similar states, and calculated the average of opinion correctness over similar states. This resulted in about 5.5K data rows of the form: (state, average opinion correctness). The average opinion correctness of a state is then used as an objective metric for the confidence level. We used Hybrid neural Fuzzy Inference System (HyFIS) [17] to build the model and learn its rules. Using the training data, the trained model has 169 fuzzy rules.

V. Experiments

We set up several simulation experiments to evaluate the proposed algorithm and compare its performance to the three state-of-the-art algorithms (Voter [4], Direct comparison (DC) [5], and Majority [3]). As proposed in [3], we extend these algorithms by allowing each agent to calculate opinion concentration γ, as in equation 2. In this way, an agent can lock in a final decision when this decision becomes sufficiently popular.

The same simulation platform with the same parameter setting is used for all the algorithms to ensure fairness. Each algorithm was run 180 times: 30 iterations $\times$ 6 complexity conditions (3 feature ratios $\times$ 2 homogeneity conditions). Without loss of generality, we assume that the white tiles are always the most frequent features. In non-homogeneous environments, the starting region of the swarm was black in half of the iterations and white in the other half. The parameter setting used for the simulation is listed in table I.

VI. Results

This section presents the results of the evaluation experiments by reporting the mean and standard deviation (SD) of each performance metric. T-tests were conducted to investigate the statistical differences between pairs of algorithms. The criterion for statistical significance is set at $p = .05$.

TABLE I
THE PARAMETER SETTINGS USED IN THE EXPERIMENTS.

L_{env}	L_{nest}	N	T_{max}	T_{cycle}	R_{comm}	θ	w	λ
1.2 m	0.26 m	100	500 min	180 s	0.13 m	0.05	0.9	1/240 s^{-1}

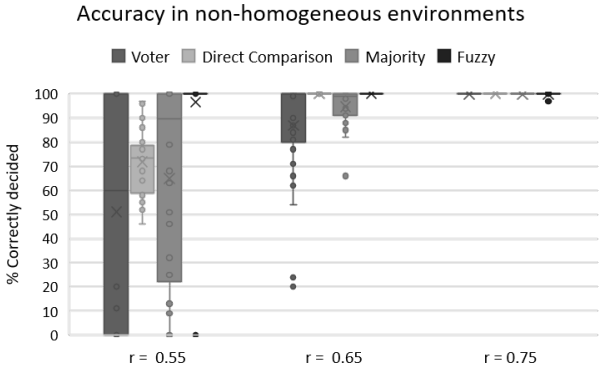

Fig. 2. The percentage of agents which reached the correct decision in non-homogeneous environments with different feature ratios.

A. Accuracy

Algorithm accuracy is evaluated in terms of the percentage of agents that reached the correct decision. In homogeneous environments, all the algorithms achieved 100% accuracy in all the simulation runs across the different levels of feature ratios (r = 0.75, r = 0.65, and r = 0.55).

On the other hand, non-homogeneous environments witnessed different levels of accuracy, as shown in figure 2. In non-homogeneous environments with high feature ratio (r = 0.75), all the algorithms achieved similar and very high levels of accuracy ranging from an average accuracy of 99.7% (by the Majority algorithm) to 100% (by the DC algorithm). When the feature ratio was medium (r = 0.65), both the proposed and the DC algorithms were 100% accurate in all the runs. However, the average accuracy of the Majority and the Voter algorithms dropped notably to 94.7% (SD = 7.5) and 86.8% (SD = 21.6), respectively. In environments with low feature ratio (r = 0.55), the fuzzy-based algorithm achieved the highest average accuracy of 96.6% (SD = 17.9). The average accuracy of the Majority, DC, and Voter algorithms dropped drastically to 64.9% (SD = 39.5), 71.8% (SD = 12.8), and 51% (SD = 49.1); respectively. Not only is a statistically significant difference found between the accuracy of the fuzzy-based algorithm and the other ones ($p \leq .0002$), but also a practically significant difference is noted as the fuzzy-based was ahead of the other algorithms by at least 24%.

B. Task time

Figure 3 shows the time taken by each algorithm to finish the task in homogeneous environments with different feature ratios. When the feature ratio was high, all the algorithms managed to finish the task within a maximum of 12 minutes. Nonetheless, the fuzzy-based algorithm needed a statistically significantly less time (mean = 5.2, SD = 1.1 minutes) than

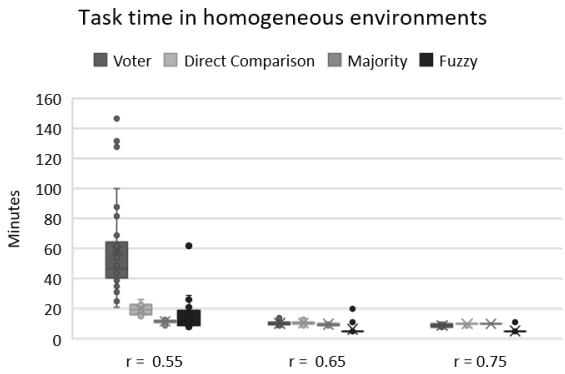

Fig. 3. The time taken by each algorithm to complete the task in homogeneous environments with different feature ratios.

all the other algorithms whose average time was over 9 minutes, ($p < .0001$). In homogeneous environments with medium feature ratio, a slight increase is recorded for the task time of the fuzzy-based algorithm (mean = 6.5, SD = 3.9 minutes). However, the fuzzy-based algorithm maintained the shortest task time that is statistically different from all the other algorithms' ($p < .0001$). Moving to environments with low feature ratio, the Majority algorithm had the lowest mean task time of 11.6 (SD = 1.6) minutes which, however, was not statistically significantly different from the task time of the proposed algorithm (mean = 15.3, SD = 10.4 minutes), ($p = .0575$). The fuzzy-based algorithm had a statistically significantly shorter task time than the DC and the Voter algorithms ($p < .0371$) and ($p < .0001$), respectively.

Non-homogeneous environments caused more salient differences in the task time, as shown in figure 4. In scenarios with high feature ratio, the fuzzy-based algorithm had the lowest mean task time, of 15.4 (SD = 4.3) minutes, which was not statistically significantly different from that of the Majority algorithm of 16.4 (SD = 3.4) minutes, ($p = .3368$). The differences between the task time of the fuzzy-based algorithm and both the DC (mean = 52.5, SD = 14.4) and the Voter (mean = 41.2, SD = 27) algorithms were larger and statistically significant, ($p < .0001$). In medium feature-ratio environments, the fuzzy-based algorithm was in the lead with a huge average difference of at least 85 minutes from the second fastest competitor, the Majority algorithm. Finally, in scenarios with low feature ratio, we found that the Voter algorithm had the lowest task time (mean = 89.3, SD = 116.6) followed by the fuzzy-based algorithm (mean = 203, SD = 76), ($p < .0001$). The Majority algorithm comes next (mean = 292.3, SD = 208.7) with statistically significant longer task times than the fuzzy algorithm ($p = .0316$). The DC algorithm had a task time of 500 minutes in all the scenarios which means that it failed to finish the task till it was forced to do so by the end of the simulation.

C. Majority rate

The algorithms are also compared in terms of their ability to reach majority. The majority rate is calculated as the ratio

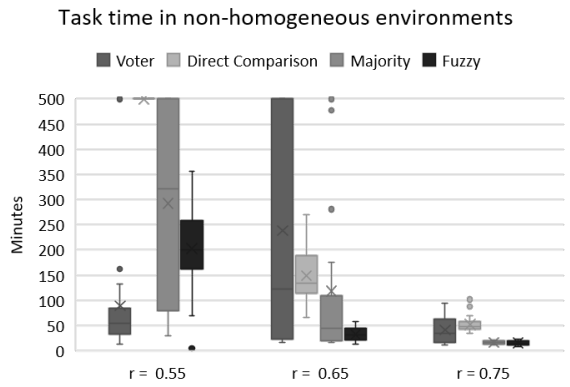

Fig. 4. The time taken by each algorithm to complete the task in non-homogeneous environments with different feature ratios.

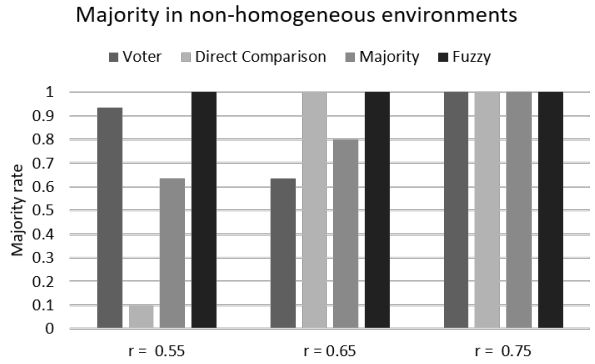

Fig. 5. The majority rate achieved by the different algorithms in non-homogeneous environments with different feature ratios.

of simulation runs in which at least 90% of the agents had the same final decision. In homogeneous environments, all the algorithms achieved a majority rate of 1, across scenarios with different feature ratios.

On the contrary, majority was not easy to achieve in some non-homogeneous environments. When the feature ratio was high, all the algorithms maintained the majority rate of 1. However, in environments with medium feature ratio, only the fuzzy-based and DC algorithms maintained their perfect majority rate while the Majority and the Voter algorithms recorded rates of 0.8 and 0.63, respectively. Finally, when the feature ratio was low, only the fuzzy-based algorithm maintained the majority rate of 1. Meanwhile, the rates for the Voter, Majority, and DC algorithms are 0.93, 0.63, and 0.1; respectively.

Table II summarises the results described in this section. The statistical difference between the proposed algorithm and the three state-of-the-art algorithms is tested. It can be seen that the proposed algorithm achieves the overall best performance as measured by the mean values of each performance metric averaged over all the 180 scenarios.

TABLE II
SUMMARY OF THE COMPARISON BETWEEN THE DIFFERENT ALGORITHMS. MEAN VALUES OF THE PERFORMANCE METRICS, AVERAGED OVER ALL THE SCENARIOS, ARE REPORTED WITH THE STANDARD DEVIATION WRITTEN BETWEEN PARENTHESES. STATISTICAL SIGNIFICANCE IS INDICATED BY * AND *** TO REPRESENT SIGNIFICANCE LEVELS OF $p < .05$ AND $p < .001$, RESPECTIVELY.

Algorithm	Accuracy	Task time	Majority rate
Voter	89.6% (28.3) ***	74.6 (131.1) *	0.93 (0.07) ***
DC	95.3% (11.8) ***	123.6 (176.3) ***	0.85 (0.13) ***
Majority	93.2% (20.8) ***	76.4 (149.2) *	0.91 (0.04) ***
Fuzzy	99.4% (7.4)	46.3 (77.5)	1 (0)

VII. CONCLUSIONS AND FUTURE WORK

In consensus achievement algorithms, agents that are highly confident in their opinions are given higher chances to influence the decision making than less confident agents. In the state-of-the-art algorithms, an agent uses its estimated feature strength to evaluate the level of confidence in its opinion. This may result in scenarios which degrades the decision making process where the most confident agents are those with incorrect opinions. This is particularly the case in non-homogeneous environments where feature distribution varies across different regions. In such environments, the performance of the state-of-the-art algorithms suffers dramatically. To avoid this problem, we proposed a fuzzy-based approach for the objective assessment of agents' confidence in their opinions. The objectivity of the estimated confidence stems from the fact that agents use previous experience to learn the mapping from states to expected opinion correctness. We characterised agent states in terms of four variables that carry complementary information about the observations made by the agent as well as the perceived opinions of the other agents. An FIS is used by agents to map their states to confidence levels.

The results of the experiments demonstrate the pronounced merits of the proposed algorithm over all the rival ones. Our algorithm could beat the other algorithms by achieving statistically and practically significant improvements in the performance as measured by accuracy, task time, and majority rate. Even in the most complex environments (with low feature ratio and non-homogeneous feature distribution), the proposed algorithm maintained its high performance by achieving an average accuracy of 96.6% while recording the lowest increase in task time. Technically, the Voter algorithm had lower task times in the most complex environments, but it completely fails in these scenarios as its accuracy drops to about 50%.

It is worth mentioning that performance gains achieved by our algorithm did not require agents to exchange new pieces of information. Messages containing only agents' ID and binary

opinion are used in our algorithm. While using non-binary opinion representation could have further improved the performance, we used binary opinion representation similar to the state-of-the-art algorithms as exchanging non-binary opinions would require more network resources. That is, we did not want to compromise communication cost for algorithm performance when comparing between the proposed and the state-of-the-art algorithms. The power of our proposed approach, however, can be attributed to two factors: 1- defining states in terms of the four variables presented in subsection IV-C and 2- using experience to learn the mapping between states and confidence levels. This results in a reliable confidence estimation, where the confidence level approximates the actual reliability of being correct. Consequently, agents that have correct opinions are given a higher chance to influence the conclusion.

In this work, the performance of the proposed approach was evaluated in a simulation platform. To compare its performance with the state-of-the-art algorithms, all the algorithms were tested on the same simulation platform using the same agent's capabilities (e.g communication range, locality of observations, and velocity). Nonetheless, we consider evaluating our approach in physical environments using real robots. This will help evaluate the proposed approach more comprehensively and support its validity.

Similar to past studies, the environmental feature used in this work is binary. As a future work, we consider evaluating the performance of our algorithm in environments with non-binary features. Another future direction is to extend the proposed algorithm so that the swarm members become able to estimate the feature ratio r rather than just deciding which feature value is the most prevalent.

REFERENCES

[1] G. Valentini, E. Ferrante, and M. Dorigo, "The best-of-n problem in robot swarms: Formalization, state of the art, and novel perspectives," *Frontiers in Robotics and AI*, vol. 4, p. 9, 2017.

[2] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, pp. 1–41, jan 2013.

[3] J. T. Ebert, M. Gauci, and R. Nagpal, "Multi-feature collective decision making in robot swarms," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1711–1719, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[4] G. Valentini, H. Hamann, and M. Dorigo, "Self-organized collective decision making: The weighted voter model," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 45–52, International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[5] G. Valentini, D. Brambilla, H. Hamann, and M. Dorigo, "Collective perception of environmental features in a robot swarm," in *International Conference on Swarm Intelligence*, pp. 65–76, Springer, 2016.

[6] G. Valentini, H. Hamann, and M. Dorigo, "Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1305–1314, International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[7] C. Lee, J. Lawry, and A. Winfield, "Combining opinion pooling and evidential updating for multi-agent consensus," International Joint Conferences on Artificial Intelligence, 2018.

[8] G. Cai and D. Sofge, "An urgency-dependent quorum sensing algorithm for n-site selection in autonomous swarms," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1853–1855, International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[9] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 541–549, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[10] A. Couture-Beil, R. T. Vaughan, and G. Mori, "Selecting and commanding individual robots in a multi-robot system," in *2010 Canadian Conference on Computer and Robot Vision*, pp. 159–166, IEEE, 2010.

[11] M. A. M. de Oca, E. Ferrante, A. Scheidler, C. Pinciroli, M. Birattari, and M. Dorigo, "Majority-rule opinion dynamics with differential latency: a mechanism for self-organized collective decision-making," *Swarm Intelligence*, vol. 5, no. 3-4, pp. 305–327, 2011.

[12] A. Bounceur, M. Bezoui, U. Noreen, R. Euler, F. Lalem, M. Hammoudeh, and S. Jabbar, "Logo: A new distributed leader election algorithm in wsns with low energy consumption," in *International Conference on Future Internet Technologies and Trends*, pp. 1–16, Springer, 2017.

[13] D. Tarapore, A. L. Christensen, P. U. Lima, and J. Carneiro, "Abnormality detection in multiagent systems inspired by the adaptive immune system," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 23–30, International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[14] A. Giusti, J. Nagi, L. M. Gambardella, and G. A. Di Caro, "Distributed consensus for interaction between humans and mobile robot swarms," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pp. 1503–1504, International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[15] J. Nagi, A. Giusti, L. M. Gambardella, and G. A. Di Caro, "Human-swarm interaction using spatial gestures," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3834–3841, IEEE, 2014.

[16] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338 – 353, 1965.

[17] J. Kim and N. Kasabov, "Hyfis: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems," *Neural Networks*, vol. 12, no. 9, pp. 1301–1319, 1999.