Quantification and Analysis of the Resilience of Two Swarm Intelligent Algorithms

Conference Paper · October 2017

DOI: 10.29007/5fhn

CITATIONS
0

READS
83

4 authors:

Joshua Cherian Varughese
Graz University of Technology
9 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Ronald Thenius
Karl-Franzens-Universität Graz
69 PUBLICATIONS **750** CITATIONS
SEE PROFILE

Thomas Schmickl
Karl-Franzens-Universität Graz
196 PUBLICATIONS **1,981** CITATIONS
SEE PROFILE

Franz Wotawa
Graz University of Technology
369 PUBLICATIONS **2,800** CITATIONS
SEE PROFILE

Quantification and Analysis of the Resilience of Two Swarm Intelligent Algorithms

Joshua Cherian Varughese[1,2], Ronald Thenius[1], Thomas Schmickl[1], and Franz Wotawa[2]

[1] Institut für Zoologie
Karl Franzens Universität Graz, Austria
joshua.varughese@uni-graz.at
[2] Institut für Software Technologie
Technische Universität Graz, Austria

Abstract

Nature showcases swarms of animals performing various complex tasks efficiently where capabilities of individuals alone in the swarm are often quite limited. Swarm intelligence is observed when agents in the swarm follow simple rules which enable the swarm to perform certain complex tasks. This decentralized approach of nature has inspired the artificial intelligence community to apply this approach to engineered systems. Such systems are said to have no single point of failure and thus tend be more resilient. The aim of this paper is to put this notion of resilience to the test and quantify the robustness of two swarm algorithms, namely "swarmtaxis" and "FSTaxis". The first simulation results of the effects of introducing an impairment in agent-to-agent interactions in these two swarm algorithms are presented in this paper. While the FSTaxis algorithm shows a much higher resilience to agent-to-agent communication failure, both the FSTaxis and swarmtaxis algorithms are found to have a non-zero tolerance towards such failures.

1 Introduction

Swarm intelligence is a well studied phenomenon in bees [9], fireflies [5], fish [2] etc. One of the most fascinating aspects of swarming behavior is their high tolerance towards loss of individual entities without losing the overall performance of the superorganism. Another noticeable aspect of the system is that swarm entities with very limited abilities work together to perform a much more complex task in a decentralized manner. If successfully implemented in robotics or similar systems, this will similarly lead to the ability of the system to function resiliently. Inspired by natural swarms and its resilience, much effort has been directed at designing systems in a decentralized and self organizing manner for higher resilience and flexibility [1].

Several studies have been conducted in the past about the aforementioned robustness of swarm intelligent algorithms. In [8], the authors present an analysis of the BEECLUST algorithm and experimentally verify the robustness of the algorithm by adding agents with impaired temperature sensors. In [4], the authors perform a comprehensive analysis of the resilience of the swarmtaxis algorithm with respect to failure modes such as IR sensor failure and motor failure. For the purpose of this paper, we consider the ability of a system to perform its overall goals even with suboptimal agent behavior to be its "resilience". In order to approach the topic of resilience of the algorithms in a comprehensive manner, we need to consider all the capabilities of the individuals of a swarm and how their failure could affect the overall goals of the swarm. Broadly speaking about functional components in any robotic system, they can be categorized into communication systems, sensors and actuators. Failure of each of these systems can affect swarms in different ways. In this paper, we will concentrate on agent-to-agent communication failures and its effects on the overall swarm behavior.

This study is part of project subCULTron [13] which aims at developing a swarm of autonomous underwater robots to perform environmental measurements and monitoring. One of the subtasks that has been identified is the swarm being able to follow a gradient. This motivation leads the authors to investigate swarm intelligent behaviors which can be used to navigate a group of robots from a starting point to a predefined goal. Since the robots are operating underwater in a real world scenario, in order to avoid the loss of robots, it has to be stressed that the robots need to be connected directly or indirectly at all times. Since this paper deals with the effects of communication behavior on the overall swarm performance, we have to clarify what kind of communication is available on the robotic platforms we are working with. Classic long range underwater communication is mainly based on acoustics. However, acoustics are expensive and susceptible to interference and cross talk especially when a swarm of robots need to communicate with one another. Therefore, we will use a local communication method, blue-light communications, where we exchange small packets of modulated blue-light signals. The range of such a communication device has been tested to be around one meter under water. Keeping the limited communication bandwidth and also the possibility of loss of packets, our algorithms and tasks must be resilient to failure in agent-to-agent communication. The rest of the paper is dedicated to the selection and resilience test of two swarm intelligent algorithms.

In the following sections of the paper, we will briefly describe how the algorithms under consideration work (section 2), then in the section on methods (Section 3), we will establish a method by which we will simulate failures of the communication devices and also define performance parameters. Subsequently, we will present the results (Section 4) and discuss them (Section 5) before concluding the paper (Section 6).

2 Algorithms

Algorithms for collective navigation and foraging problems are the most investigated fields in swarm intelligence [15][6][11]. These algorithms that are inspired by natural swarms, can be broadly subdivided into pheromone based navigation [14], navigation based on physical robot chains [19], navigation based on signaling [3] [17] and navigation based on pheromone-like gradients [10] [7] which can be considered to be a combination of navigation based on explicit signaling, pheromones, and robot chains. In [14], the authors use a combination of cameras and LEDs to project virtual pheromone trails which “evaporate” with time. Such global observation based algorithms are evidently not suitable for an underwater environment where global observation is expensive and difficult to implement. In [19], the authors use immobile robots as “chains” from the “food” to the “nest” which is a navigation algorithm based on physical robot chains. Considering that the range of blue-light communication mechanism is low, a large number of robots will be needed to form ”beacons” which would then act as a navigation landmarks for moving robots. Therefore, such an implementation is also infeasible for subCULTron and similar underwater projects. In the approach used by [10] and [7], the authors present algorithms in which a value is exchanged between robots and this value acts as a gradient which enables the robots to navigate between the ‘food source” and the “nest”. This can be viewed as a combination of using explicit signaling and physical robot chains as a means to emulate the function of a pheromone trail. This approach also has disadvantages with respect to scalability as it would need a large number of robots for a longer trail. In the taxis approach presented in [3] and [17], the authors use a single bit ping as a signaling mechanism in their algorithms to achieve gradient taxis and source localization respectively. This approach seems to be the most suitable for underwater environments where robots need to stay cohesive

and connected. Henceforth, we will be concentrating on approaches which employ a single ping communication between agents.

As discussed in Section 1, the underwater environment introduces additional constraints over and above the widely accepted swarm intelligence criteria [16]. These additional constraints are as follows:

1. Algorithms must involve cohesive movement of agents.
2. Algorithms must involve purely local communication.
3. Algorithms must aim at navigating a swarm from a starting point to the goal.

The swarmtaxis algorithm [3] and the FSTaxis [17] algorithm are two algorithms which fulfill all of the above criteria. They are similar to each other in some aspects which include the utilization of a single ping, purely local communication etc. The main difference between these algorithms is that the swarmtaxis algorithm uses a technique that forces all the agents to be connected to its neighbors while FSTaxis demands no such constraint. Despite few differences, largely, the algorithms are comparable to each other thus enabling us to use the performance parameters we designed for this comparison. In this section, we will look into each of these algorithms in detail.

2.1 The swarmtaxis algorithm

The swarmtaxis algorithm [3] uses a differential movement to navigate a swarm to the goal. Each of these agents is equipped with a communication device to emit a single bit, a long range sensor to sense the goal, a local communication device to sense the pings (single bit communication) emitted by other agents and an avoid sensor to detect its surroundings. The "source" or "goal" can be occluded from one agent by another agent. If an agent is occluded by another agent from the source, it is said to be in "shadowed" mode; otherwise, it is in "illuminated" mode. The "avoid radius" of those agents in shadowed mode is lesser than those in illuminated mode. This means that an agent in illuminated mode can detect agents in shadowed before the latter can detect the former. While implementing the swarmtaxis algorithm, each agent is allowed to be in one of the following states: "forward","coherence", "avoid" or "random". All agents are set to the "forward" state by default and each agent chooses one of the other states depending on certain conditions. When the agent detects a drop in the number of locally connected agents below the entire population of the swarm, it enters the "coherence" state. Alternatively, when the agent detects a rise in the number of connected agents, it enters the "random" state. When an agent detects another agent within its "avoid radius", it enters the avoid state. The behavior of agents in each of these states is as follows:

1. "Forward" state: The agent moves straight ahead at a constant speed.
2. "Coherence" state: The agent executes a 180 ° turn and then enters into the "forward" state.
3. "Random" state: The agent takes a random turn and then enters into the "forward" state.
4. "Avoid" state: The agent takes a turn in the opposite direction with respect the agent it detected within its avoid radius and then enters the "forward" state.

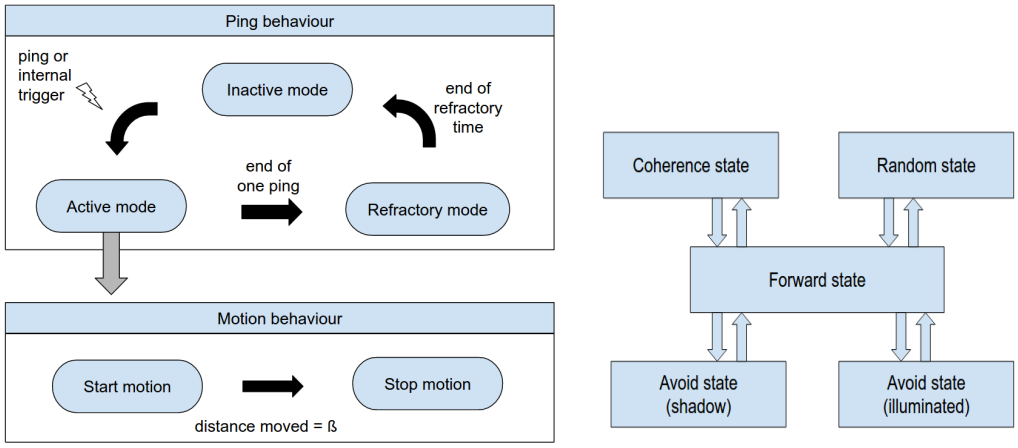
2.2 The FSTaxis algorithm

The Firefly and Slime mold Taxis (FSTaxis) algorithm [17] is an emergent gradient taxis algorithm inspired by the communication strategy used by slime mold and fireflies. In the FSTaxis algorithm, each agent has three communication states: "pinging", "refractory" and "inactive". Each of the agents has an internal countdown timer whose value is associated with its position in the environmental gradient and each agent has basic motoring capabilities. During the execution of the algorithm, all agents are set to inactive mode. In the inactive mode, the agent only checks for incoming single bit local communication (pings). When an agent receives a ping or the agent's internal timer counts down to zero, it broadcasts a ping (or 1 bit communication) for a certain duration, say t_p. Immediately as the agent receives a ping, apart from relaying the ping, the agent moves towards this incoming ping. After t_p, the agent enters the refractory mode. During refractory time, t_r, the agent ignores all incoming pings. After the refractory time, the agent sets itself back to inactive mode. The cycle continues when the agent receives another ping or when its internal timer counts down to zero before it gets a ping.

The above mentioned behavior results in "scroll waves" [12] propagating through the swarm. Since pinging can be caused by either an agent receiving a ping or by an agent's internal timer counting down to zero, the agents with low timer values will hijack the communication frequency of the entire swarm and force the swarm to ping at hijacker's frequency and also forces the rest of the swarm to move towards the hijacker. This movement results in a collective gradient ascent or descent, depending on the scaling of the gradient value. Thus, gradient taxis is a result of an emergent tendency of the swarm to move against the scroll waves emerging from single bit communications between agents. A typical run of the FSTaxis algorithm in simulation is shown in Figure 2(b). For more details on FSTaxis, please refer to the paper on the FSTaxis algorithm [17].

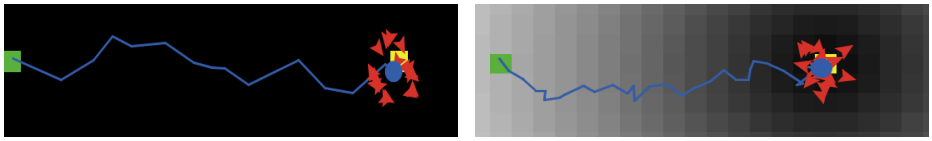
It is clear from the description above that the communication module needed in the FSTaxis algorithm is only the device needed to send and receive pings. This communication module on each agent is responsible for communicating a single ping to the surrounding agents. In Section 3.1, we will discuss how we simulate failure of such a communication device.

This algorithm enables a group of agents to move together towards the goal (or source). The goal in the case of the swarmtaxis algorithm is a quantity that can be measured by means of a long range sensor. For example, a light source can be the goal for the swarmtaxis because it can be measured using a long range sensor and can be occluded from some agents by other agents. Since the illuminated agents see the shadowed agents before the latter can see the former, this results in the illuminated agents moving away from shadowed agents which is in fact, the direction of the goal. A typical run in the swarmtaxis algorithm is shown in Figure 2(a). As discussed above, the "coherence" state of the swarmtaxis algorithm keeps the agents together. After the initial publication [3], the authors presented a method where the entire swarm needed to be connected for the swarm to consistently move forward towards the goal without entering the "coherence" state. In later modifications of the swarmtaxis algorithm [21][4], the authors presented improved version (β and ω versions) of the algorithm, where the agents were required to communicate more than a single ping in order to reliably work. In this paper, we will therefore consider the basic algorithm presented in [3], and based on the connectivity study presented in [21], we will present the improvement in resilience due to the relaxation of the connectivity constraint α.



(a) (b)

Figure 1: State transition diagrams of both algorithms are shown in the figures. Figure 1(a) shows the state transition diagram of the FSTaxis algorithm and Figure 1(b) shows the state transition diagram of the swarmtaxis algorithm [3]



(a) (b)

Figure 2: Typical runs of the swarmtaxis and the FSTaxis algorithms are shown in Figures 2(a) and 2(b), respectively. The green patch represents the starting point (randomly chosen), the yellow patch represents the goal, the blue trace represents the trajectory of the centroid (blue circle) of the swarm. The red arrow like shapes at the goal represent agents and the patch colors in Figure 2(b) represent the local gradient value.

3 Methods

In this section, we describe how we simulated[1] failures in the algorithms of interest and also which performance measures we used to study the effects of agent-to-agent communication failure in each of these algorithms. The parameters used for simulation are the same as those used in [17] and [3] for FSTaxis and swarmtaxis respectively. In order to ensure fair comparison, a common communication range of 3 patches (distance unit in Netlogo) and an agent velocity of 0.5 patches per time step has been used for agents in both algorithms. The initial distribution of agents around the starting point is a uniform distribution. In order to minimize run to run differences and enable appropriate comparison, the same starting point and ending point are used for all experiments conducted henceforth for both algorithms.

[1]The simulation of both of these algorithms were done in Netlogo 5.3.1 simulation environment [20].

3.1 Simulating failures

In Section 2, we discussed briefly about each algorithm and we saw that, in both of these algorithms, agents use a single bit communication method to let the surrounding agents know of their presence. A failure in the communication device would mean that the other agents will not detect the presence of the failed agent. The illustration of such a case in the FSTaxis algorithm and in the swarmtaxis algorithm is shown in figures 3 and 4 respectively. In order to simulate this failure, we used a probability based roll of a dice each time the agent attempts to communicate and decided whether the communication should fail or not. Then, we increased this failure probability progressively and, for each failure probability, we collected 100 data sets in order to have substantial data to support our conclusions. In the following subsection, we discuss data collection during each simulation run and why each parameter is suitable to analyze the performance of the swarm.

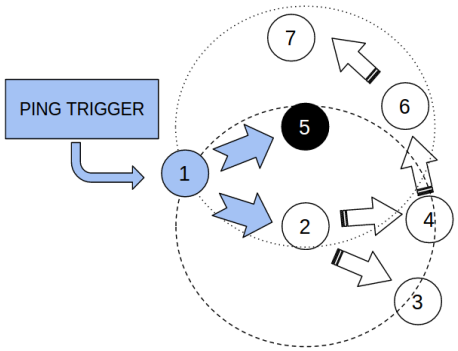


Figure 3: A scenario of ping failure in the FSTaxis algorithm. Agent 1, triggered by its internal counter, broadcasts a ping. The arrows represent the relaying of that ping to those agents whose communication device can detect this ping (represented by dotted circles for agents 5 and 2). Agent 2 is normal and relays the ping to the nearby agents. Agent 5 (black color) has a malfunction in its communication module and hence does not relay the ping to agents 6 and 7. Agent 6 gets the ping via agent 4 and that ping is in turn relayed to agent 7.

3.1.1 Ping failure in FSTaxis

Figure 3 illustrates an event of ping failure in the FSTaxis algorithm. Agent 1, triggered by its internal counter, broadcasts a ping. The arrows represent the relaying of that ping to the agents whose communication device can detect this ping. Agent 2 is normal and relays the ping to the surrounding agents. Agent 5 (black color) has a malfunction in its communication module and hence does not relay the ping to agents 6 and 7. This makes agent 5 invisible to the other agents in the surroundings and in effect, the "original" ping direction is misunderstood by agent 6. The result is that Agent 7 does not move at all, while agent 6 incorrectly moves towards agent 4.

3.1.2 Ping failure in swarmtaxis

Figure 4 illustrates two scenarios of ping success and failure in the swarmtaxis algorithm. In Figure 4(a), a successful scenario of pinging agents within a range is shown. The circle

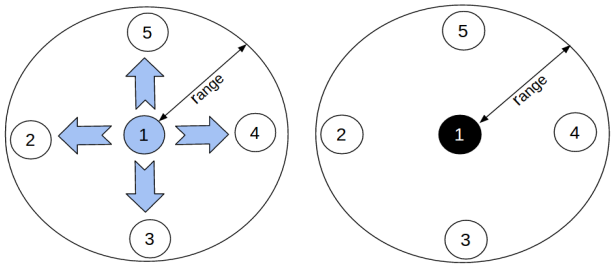


Figure 4: Two scenarios of communication in the swarmtaxis algorithm. Scenario 1 is shown on the left, where agent 1 broadcasts its periodic ping and all other agents in the range will perceive this ping. Scenario 2 is shown on the right, where agent 1 (black) has a ping malfunction which prevents it from broadcasting the ping, and hence the other agents in range are blind to the presence of agent 1.

represents a range within which all agents can communicate with each other. In Figure 4(b), a failed scenario of pinging is shown. Agent 1 (black) has a malfunction and hence does not broadcast the ping to the other agents in range. Since the swarmtaxis algorithm is based on counting the number of connected agents, a failed ping means that the other agents in range (agents 2, 3, 4, 5) register a decrease in number of connected agents.

3.2 Performance measures

In this subsection, we describe several observer level performance parameters which can be used to quantify the resilience of each algorithm as well as to compare the algorithms with each other.

3.2.1 Time performance

One of the intuitive performance measures that points directly to the performance of the swarm is the time or number of iterations the swarm takes to "converge" to the goal. We define "convergence" as the centroid of the swarm reaching the goal. The"time to convergence" for different probabilities of failure is measured for comparison with the other runs. This is a very intuitive way of penalizing runs which take longer than the baseline time for the respective algorithm. A typical run with no communication failure takes on average 2,000 iterations for the FSTaxis algorithm and 10,000 iterations for the swarmtaxis algorithm. The standard deviation of the number of iterations for each of these algorithms does not exceed 200 iterations. The difference in the averages of iterations does not imply worse performance as the number of iterations are dependent on step size of individual agents and other parameters. During experimentation with communication failure, it is possible that the swarm never converges to the goal. In order to prevent infinite run time, keeping in mind the mean and standard deviation of the number of iterations typically needed for convergence, we limited the number of iterations to 10 times the number of iterations a swarm needs to converge to the goal with all functions intact. Thus, the iteration limit is 100,000 iterations for the swarmtaxis algorithm and 20,000 iterations for the FSTaxis algorithm. Therefore the time performance, normalized against its

own ideal performance, can thus be represented as per Equation 1.

$$time_{performance} = \frac{number\_of\_iterations}{iteration\_limit} \tag{1}$$

3.2.2 Optimal path and deviation

From Figures 2(b) and 2(a) it is evident that the typical trajectories of the centroid of the swarm (hereafter referred to as "centroid trajectory") for both algorithms do not follow a straight path from the starting point to the goal. Assuming that the optimal path is a straight line joining the starting point to the goal, experiments with faulty agent-to-agent communication modules show that the centroid trajectory has a tendency to swerve away from the optimal path. Therefore, the deviation of the centroid trajectory from the optimal path has some information about suboptimal swarm behavior. Following this logic, we consider the error between the actual path and the optimal path to be a performance measure of the swarm. Figure 5 shows an illustration of deviation of a centroid trajectory from the displacement vector. Here, "start" block represents the starting point $\overrightarrow{S} = (x_s, y_s)$ of the swarm and "goal" block represents the goal $\overrightarrow{G} = (x_g, y_g)$. The free drawn line traces the actual trajectory τ of the centroid of the swarm and the ideal path can be represented as a vector $\overrightarrow{D}$. The trajectory τ can be represented as set T of point vectors in cartesian coordinates that the centroid of the swarm passed through during the actual runs. For each failure probability p_k, we conducted 100 runs and obtained the set of all centroid trajectories, O_k, and, for each run, we obtained a set τ_{kj} which contains points T_{kji}. τ_{kj} corresponds to the trajectory in the k^{th} failure probability and j^{th} run and T_{kji} corresponds to one point in the centroid trajectory of the i^{th} iteration in the j^{th} run with k^{th} failure probability. Subsequently, the projection of the point vector D_{kji} of the point vector T_{kji} on $\overrightarrow{D}$ was computed. Furthermore, the error vectors, obtained as shown in Equation 5, can be used to represent the deviation from the optimal path. From all the computed error vectors, the root mean square error E_{kj}^{rms} can be obtained across a single run from start to goal as per Equation 6. E_{kj}^{rms} represents a window of operation for the centroid trajectory of the swarm for runs with a certain p_k. The optimal window of operation is the range of E_{0j}^{rms} for all runs with zero probability of failure. Therefore, the set of all E_{kj}^{rms} for a particular p_k, say ε_k, as formulated in Equation 7 can be represented on a box-plot to visualize how the window of operation shifts with changing p_k.

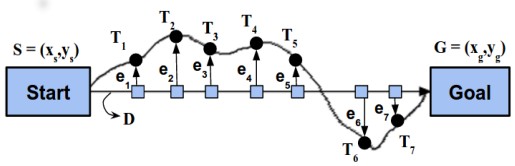


Figure 5: Illustration of the deviation of a swarm centroid trajectory from the straight line passing through (x_s, y_s) and (x_g, y_g). The "start" block represents the starting point of the swarm and "goal" block represents the end point of the swarm. The free drawn line represents the actual trajectory 'T' of the swarm and the displacement vector is considered as the ideal path 'D'. Points T_1, T_2 etc. represent samples from the swarm trajectory and e_1, e_2 etc. are the distance of points on T to corresponding points on D.

$$\begin{aligned} P \ &= \{p_k \mid p_k = 0.05, 0.1, 0.15 \ \ldots 1\} \\ &, \ where \ k \ = \ 1, 2, 3, \ \ldots, \ \mid P \mid \end{aligned} \tag{2}$$

$$\begin{aligned} \forall \ p_k \ \exists \ O_k, \ where \ O_k \ &= \ \{\tau_{kj} \mid \ j \ = \ 1, \ 2, \ldots, \ 100\} \\ and \ \tau_{kj} \ &= \ \{\overrightarrow{T_{kji}} \mid \ i \ = \ 1, \ 2, \ldots, \ N_{kj}\} \end{aligned} \tag{3}$$

$$\forall \ \overrightarrow{T_{kji}} \ \exists! \ \overrightarrow{D_{kji}} s.t. \quad (\overrightarrow{T_{kji}} - \overrightarrow{D_{kji}}) \ \perp \ (\overrightarrow{D_{kji}} - \overrightarrow{S}) \tag{4}$$

$$\overrightarrow{e_{kji}} \ = \ \overrightarrow{D_{kji}} \ - \ \overrightarrow{T_{kji}} \tag{5}$$

$$E_{kj}^{rms} \ = \ \frac{1}{N_{kj}} \sum_{i=1}^{N_{kj}} \ \|\overrightarrow{e_{kji}}\| \tag{6}$$

$$\forall \ p_k \ \exists \ \varepsilon_k, \ where \ \varepsilon_k \ = \ \{E_{kj}^{rms} \mid \ j \in [1, \ 2, \ldots, \ 100]\} \tag{7}$$

$$Q \ = \ \{\varepsilon_k \mid \ k \in [1, \mid P \mid] \ and \ k \in \mathbb{N}\} \tag{8}$$

4 Results

In order to minimize run to run differences, the same starting point $\overrightarrow{S}$ and goal $\overrightarrow{G}$ are used for all 100 runs for each p_k. Also, agents in both algorithms move with the same individual step size = 0.5 units, where one unit is unit length in cartesian coordinates.

4.0.3 Time performance

Figure 6 shows the performance parameter $t_{performance}$ and how it changes as the probability of failure increases. We see that $t_{performance}$ saturates as maximum allowed iterations are reached. Time performance saturates rapidly for the swarmtaxis algorithm, while the FSTaxis shows a wider range of operation before $t_{performance}$ saturates.

4.0.4 Root mean square error

Figures 8 and 9 show the distribution of mean square errors (E^{rms}) of the centroid trajectory with respect to the optimal path 'D' of a swarm executing the FSTaxis algorithm and the swarmtaxis algorithm respectively. We see that as the probability of failure increases, the median and spread of E^{rms} increases for the FSTaxis algorithm, while it remains more or less constant for the swarmtaxis algorithm. It is also important to note the "outliers" in the box-plot as they also contribute to the increasing spread of the distribution. The increasing deviation from the optimal path, or, in other words, the increasing range of root mean square error, shows the increasing deviation of the swarm centroid from the optimal path. This means that the "window of operation" (as defined in Section 3) widens as the ping loss increases for the FSTaxis algorithm. Those probabilities with more than 50% non-converging runs are marked as "non-converging runs".

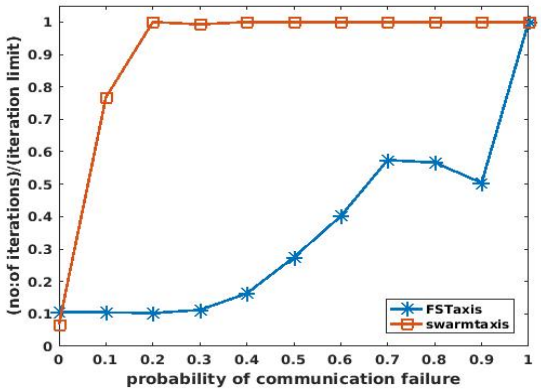


Figure 6: Performance parameter, $t_{performance}$ and how it changes as the probability of failure increases. The presented data is based on 1000 simulation runs (100 runs per p_k).

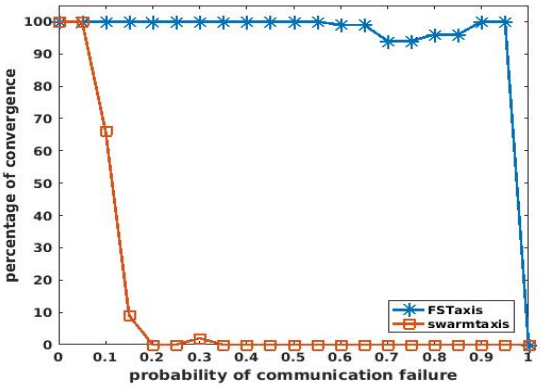


Figure 7: Percentage of runs of each algorithm that converged to the goal with increasing probability of failure. The presented data is based on 1000 simulation runs (100 runs per p_k).

5 Discussion

From figures 7 and 6, we observe a "Resilient operating limit"(ROL) of failure probabilities p_k. In Figure 7, the percentage of runs of each algorithm that converged to the goal for the FSTaxis algorithm is consistently 100% until $p_k = 70\%$, while the swarmtaxis algorithm tolerates only 5% failure probability for 100% convergence. Therefore, for the FSTaxis algorithm, ROL of p_k= 70%, while for the swarmtaxis algorithm, ROL of $p_k = 5\%$. The reason for this limited ROL of the swarmtaxis algorithm is that the swarmtaxis algorithm needs all the members of the swarm to be connected to each other in order to avoid losing swarm members. In the paper [3], we see that the α value (connected swarm members) is set to the population of the swarm. This constraint makes the swarm enter repeatedly into "coherence" state which drives the swarm away from the goal. As the probability of failure increases, the probability of at least one agent not being connected to the rest of the swarm increases drastically and, hence the swarm remains in coherence state. In contrast, the reason for high resilience of the FSTaxis algorithm

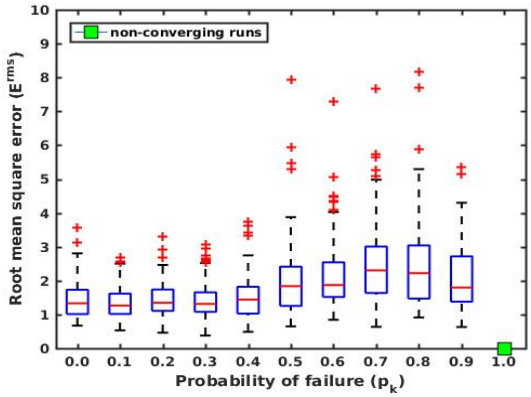


Figure 8: Distribution of root mean square error of centroid trajectory of a swarm executing FSTaxis. Those probabilities of failure with more than 50% non-converging runs are marked as "non-converging runs". The red '+' signs represent the root mean square error of a single run among the total 1000 (100 runs per p_k) runs.

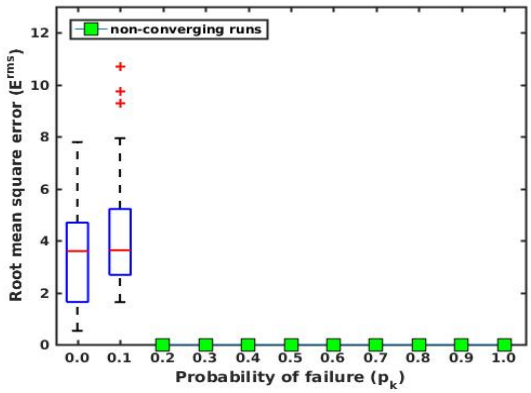


Figure 9: Distribution of root mean square error of centroid trajectory of a swarm executing swarmtaxis. Those probabilities of failure with more than 50% non-converging runs are marked as "non-converging runs". The red '+' signs represent the root mean square error of a single run among the total 1000 (100 runs per p_k) runs.

is that the behavior of the swarm is not based on the number of pings received, but rather on the presence of a ping. Even if a high amount of pings are lost due to agents failing to relay it further, due to the presence of some incoming pings, the agents move towards it. This is not always the correct direction for successful gradient taxis, but nevertheless has some information about the gradient due to the sole origin of pings being from the agent at the local gradient extrema. Therefore, the swarm takes many unnecessary steps swinging away from the optimal path, but still it manages to reach the goal each time even with a very high ping loss. In [21] and [4], the authors present a modified swarmtaxis algorithm and experiment with various α values, that is, they relax the connectivity criteria presented in [3]. Such an approach intuitively

will reduce the number of transitions into the "coherence" state and therefore slightly improve the ROL but the basic state transition is still based on a polling method.

From Figure 8, it can be inferred that the median of root mean square error of the FSTaxis algorithm increases with increasing probability of failure. A swarm operating with high probability of failure p_k has a very irregular movement around the optimal path in contrast to the exemplary run of the FSTaxis algorithm shown in Figure 2(b). This is due to the fact that pings that originate from the agent whose internal trigger counts out are lost as shown in Figure 3. As a result, the agents in the swarm either do not get the pings or get the pings via other agents whose communication mechanism is working. This in turn causes the agents to move towards the incoming ping and hence, in a suboptimal direction as compared to the goal. These suboptimal movements explain the increase in spread of root mean square error signified by the median and quartile shift. We also see more outlying data points (Figure 8) as compared to the runs with lower probability of failure due to the same reason mentioned above. It is remarkable that even with a very high amount of ping loss, the swarm manages to find the goal most of the time as shown in Figure 7.

6 Conclusion

From the above sections, evidently, the FSTaxis algorithm exhibits resilient behavior even with a high ping loss. The reason for the high resilience of the FSTaxis algorithm is the ping relaying mechanism (borrowed from slime mold and fireflies) as described in Section 2. This mechanism increases the probability of pings being relayed reliably despite ping loss. The communication mechanism of FSTaxis can be implemented in other swarm robotic algorithms or even for multi-robot systems where communication is crucial to goal achievement.

The operating range of the swarmtaxis algorithm is cut short to a resilient operating range - $0 < p_k < 0.05$ due to the algorithm repeatedly driving the swarm into "coherence" mode. In [21], the authors present various techniques to ensure connectivity and avoid unnecessary state transitions into"coherence" state. However, these modifications do not retain the 1-bit communication feature which is a strong argument for underwater swarms since communication is expensive and subject to noise.

For projects like subCULTron which aim at developing algorithms in environments where communication is costly, the FSTaxis can be employed for its resilience. Since the behavior of swarmtaxis algorithm is not tightly coupled with the polling mechanism it employs, one very interesting question at this point is "What would be the result if one were to implement the communication strategy of the FSTaxis algorithm in swarmtaxis?". The authors have published [18] a modified swarmtaxis algorithm to employ the communication method of FSTaxis based on the research presented in this paper.

7 Acknowledgements

This work was supported by EU-H2020 Project no. 640967, subCULTron, funded by the European Unions Horizon 2020 research and innovation programme.

References

[1] Andreas Angerer, Michael Vistein, Alwin Hoffmann, Wolfgang Reif, Florian Krebs, and Manfred Schnheits. Towards multi-functional robot-based automation systems. In *Proc. 12th Intl. Conf.*

on Inform. in Control, Autom. & Robot., Rome, Italy, 2015.

[2] I. Aoki. A simulation study on the schooling mechanism in fish. *Bulletin of the Japanese Society of Scientific Fisheries*, 48(8):1081–1088, 1982.

[3] Jan Dyre Bjerknes, Alan Winfield, and Chris Melhuish. An analysis of emergent taxis in a wireless connected swarm of mobile robots. In *IEEE Swarm Intelligence Symposium*, pages 45–52, Los Alamitos, CA, 2007. IEEE Press.

[4] Jan Dyre Bjerknes and Alan FT Winfield. On fault tolerance and scalability of swarm robotic systems. In *Distributed autonomous robotic systems*, pages 431–444. Springer, 2013.

[5] John Buck and Elisabeth Buck. Biology of synchronous flashing of fireflies. *Nature*, 211:562–564, 1966.

[6] Frederick Ducatelle, Gianni A Di Caro, Alexander Förster, Michael Bonani, Marco Dorigo, Stéphane Magnenat, Francesco Mondada, Carlo Pinciroli, Philippe Rétornaz, Vito Trianni, Luca M Gambardella, F Ducatelle, Ga Di Caro, A Förster, LM Gambardella, M Bonani, S Magnenat, F Mondada, P Rétornaz, M Dorigo, C Pinciroli, and V Trianni. Cooperative navigation in robotic swarms. *Swarm Intell*, 8:1–33, 2014.

[7] Nicholas R Hoff, Amelia Sagoff, Robert J Wood, and Radhika Nagpal. Two foraging algorithms for robot swarms using only local communication. In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, pages 123–130. IEEE, 2010.

[8] Daniela Kengyel, Payam Zahadat, Thomas Kunzfeld, and Thomas Schmickl. Collective decision making in a swarm of robots: How robust the beeclust algorithm performs in various conditions. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS)*, BICT'15, pages 264–271, ICST, Brussels, Belgium, Belgium, 2016. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[9] Charles Duncan Michener. *The social behavior of the bees: a comparative study*, volume 73. Harvard University Press, 1974.

[10] T. Schmickl and K. Crailsheim. Trophallaxis within a robotic swarm: bio-inspired communication among robots in a swarm. *Autonomous Robots*, 25(1-2):171–188, aug 2008.

[11] Madhubhashi Senanayake, Ilankaikone Senthooran, Jan Carlo Barca, Hoam Chung, Joarder Kamruzzaman, and Manzur Murshed. Search and tracking algorithms for swarms of robots: A survey. *Robotics and Autonomous Systems*, 75:422–434, 2016.

[12] Florian Siegert and Cornelis J. Weijer. Three-dimensional scroll waves organize dictyostelium slugs. *PNAS*, 89(14):6433–6437, July 1992.

[13] subCULTron. Submarine cultures perform long-term robotic exploration of unconventional environmental niches, 2015. http://www.subcultron.eu/.

[14] Ken Sugawara, Toshiya Kazama, and Toshinori Watanabe. Foraging behavior of interacting robots with virtual pheromone. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 3074–3079. IEEE, 2004.

[15] Ying Tan and Zhong-yang Zheng. Research Advance in Swarm Robotics. *Defence Technology*, 9(1):18–39, 2013.

[16] Ali E Turgut, F Gokce, Hande Celikkanat, L Bayindir, and Erol Sahin. Kobot: A mobile robot designed specifically for swarm robotics research. *Middle East Technical University, Ankara, Turkey, METU-CENG-TR Tech. Rep*, 5(2007), 2007.

[17] Joshua Cherian Varughese, Ronald Thenius, Franz Wotawa, and Thomas Schmickl. Fstaxis algorithm: Bio-inspired emergent gradient taxis. In *Proceedings of the Fifteenth International Conference on the Synthesis and Simulation of Living Systems*. MIT Press, 2016.

[18] Joshua Cherian Varughese, Ronald Thenius, Franz Wotawa, and Thomas Schmickl. swarmfstaxis: Borrowing a swarm communication mechanism from fireflies and slime mold. In *Proceedings of the Twenty First Annual Meeting on Agent Based Modeling and Simulation*. Springer, 2017. in print.

[19] Barry Brian Werger and Maja J Mataric. Robotic" food" chains: Externalization of state and program for minimal-agent foraging. In *In (Maes et al.* Citeseer, 1996.

[20] Uri Wilensky. Netlogo. *Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL*, 1999.

[21] Alan FT Winfield and Julien Nembrini. Emergent swarm morphology control of wireless networked mobile robots. In *Morphogenetic Engineering*, pages 239–271. Springer, 2012.

A Metadata for the publication

The netlogo code for the swarmtaxis and the FSTaxis algorithms is available at: https://drive.google.com/open?id=0B-VPQrRNqQZ6V0NhN0V1VFFJMkE