Joshua Cherian Varughese, MSc

Wave Oriented Swarm Paradigm

Unification of common swarm behaviors using minimalistic communication

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Univ.Prof. Dipl.-Ing. Dr. Franz Wotawa
Institute for Software Technology

Co-supervisor

Univ.Prof. Mag. Dr.rer.nat.Thomas Schmickl
Institute for Biology,
University of Graz

Graz, October 2019

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date	Signature

Abstract

Swarm intelligence and swarm robotics research have been underway for a few decades and has produced many algorithms based on self organizing systems found in nature. Contrary to classical control systems, self organizing behaviors found in nature rely on simple decentralized rules. Synchronization of a swarm of agents based on pulse coupled oscillators which are inspired by fireflies blinking in unison is one example of artificial self organizing behaviors inspired by nature. Another example of such a system is the emergent aggregation of a group of robots at a global thermal optima inspired by the aggregation of newly hatched bees. While a large body of research exists for variations and modifications in many such algorithms, there have been only few attempts to unify the underlying agent level design of these widely varying behaviors.

In this thesis, a design paradigm for a swarm of agents is presented which can exhibit a wide range of collective behaviors at swarm level. The paradigm draws inspiration from the traveling wave based chemical communication found in slime mold and the periodicity of blinking in fireflies. I demonstrate in simulation and on real robotic platforms that using this simple underlying communication paradigm at agent level, a swarm can inherit various swarm level abilities such as synchronization, leader election, aggregation, gradient ascent etc. Additionally, I also attempt to combine above mentioned behaviors to enable a programmer to perform complex behaviors at a swarm level. Adding to the novelty and versatility of the presented paradigm, a 1-bit situated communication is used throughout the thesis, which demonstrates the minimalistic nature of the presented paradigm. In addition, the resilience of the paradigm to the loss of incoming communication and how such disturbances affect the overall behavior is studied.

At the end of the thesis, a practical application of the presented paradigm is suggested for detecting an oxygen depletion phenomenon called anoxia

in the lagoon of Venice using robotic platforms developed during project subCULTron. By using the presented design paradigm, simple robots with limited capabilities can monitor the local conditions and eventually contribute to detecting a spreading phenomenon like anoxia using purely local information and communication.

Kurzfassung

Die bereits seit einigen Jahrzehnten stattfindende Forschung im Bereich Schwarmrobotik und Schwarmintelligenz führte zur Entwicklung diverser Algorithmen, die auf selbstorganisierten Systemen in der Natur basieren. Im Gegensatz zu klassischen Kontrollsystemen stützt sich selbstorganisiertes Verhalten in der Natur auf einfache, dezentralisierte Regeln. Ein Beispiel für bio-inspiriertes, selbstorganisiertes Verhalten in einem künstlichen System ist Synchronisation, basierend auf gekoppelten Oszillatoren in einem Schwarm von Agenten, welche von Glühwürmchen inspiriert sind, die im Einklang blinken. Ein weiteres Beispiel für ein solches System ist die emergente Aggregation in Gruppen von Robotern bei einem globalen thermischen Optimum, das durch das Verhalten frisch geschlüpfter Bienen inspiriert wurde. Während für Variationen und Modifikationen in vielen derartigen Algorithmen eine große Anzahl an Untersuchungen existiert, gibt es nur wenige Ansätze, die das zugrunde-liegende Design auf Agenten-Ebene dieser stark variierenden Verhaltensweisen vereinheitlichen.

In dieser Arbeit wird ein Design-Paradigma für einen Schwarm von Agenten präsentiert, welches eine breite Palette an kollektivem Verhalten auf Schwarm-Ebene erzeugen kann. Das Paradigma ist von der auf propagierenden Wellen basierenden chemischen Kommunikation inspiriert, welche einerseits in bestimmten Schleimpilzen, und andererseits in der Periodizität des Blinkens von Glühwürmchen zu finden ist. In Simulationen und mithilfe realer Roboter, die dieses einfache, grundlegende Kommunikationsparadigma auf Agenten-Ebene nutzen, wird gezeigt, dass ein Schwarm verschiedene Fähigkeiten auf Schwarm-Ebene erhalten kann, wie zum Beispiel Synchronisation, Wahl eines Anführers, Aggregation oder Bewegung entlang von Gradienten, um nur ein paar zu nennen. Darüber hinaus wird die Kombination der oben genannten Verhaltensweisen auf Schwarm-Ebene untersucht, um es ProgrammiererInnen zu ermöglichen, komplexe Verhaltensweisen auf Schwarm-Ebene unter Verwendung des vorgestellten Paradigmas zu erzeugen. Zusätzlich zur Neuheit und Vielseitigkeit des vorgestellten Paradigmas

wird in der gesamten Arbeit 1-Bit-Kommunikation verwendet, die den minimalistischen Charakter des vorgestellten Paradigmas deutlich macht. Außerdem wird die Resilienz des Paradigmas gegenüber dem Verlust eingehender Kommunikation untersucht und auch, wie sich solche Störungen auf das Gesamtverhalten auswirken.

Am Ende der Arbeit präsentiere ich eine mögliche praktische Anwendung des vorgestellten Paradigmas, nämlich die Erfassung eines anoxischen Phänomens in der Lagune von Venedig, die mit Hilfe von Schwarmrobotern erfasst wird, welche im Projekt subCULTron entwickelt wurden. Unter Verwendung des vorgestellten Design-Paradigmas können einfache Roboter mit begrenzten Fähigkeiten die lokalen Umweltparameter und Bedingungen überwachen und schließlich dazu beitragen, sich ausbreitende Phänome, wie anoxische Bereiche in der Lagune von Venedig, unter Verwendung von rein lokaler Information und Kommunikation zu erkennen.

Acknowledgment

I would like to thank my supervisor Prof. Dr. Franz Wotawa who never failed to motivate me to keep working on my PhD. Without his relentless support and guidance, I would not have made it to this point. I would also like to thank my second supervisor Prof. Dr. Thomas Schmickl for guiding me with regards to scientific work as well as project management. I am confident that the guidance I got from both my supervisors will remain crucial in my future endeavors.

A special word of thanks to Dr. Ronald Thenius whose ideas have enabled me to publish various papers. His guidance and enthusiasm is and has been an invaluable ingredient for not only my doctoral research but also for life in general. I am forever grateful for his guidance, advice, love and care that goes much beyond any call of duty. I also thank Dr. Payam Zahadat for her unwavering support throughout my time at the Artificial Life Lab.

I could not have made it to this point without the invaluable inputs in various niches from colleagues the Artificial Life Lab. Many have grown to be great friends with the slight handicap of ever having been colleagues. Hannes Hornischer, your annoying perfectionism has played a great role in making me a better researcher. Daniel Hofstadler, your quick wit and you being a pundit will remain a refreshing memory. Daniel E. Moser, your friendship has carried me through many a difficult days, tiresome workshops and late night soldering sessions. The widely varying contributions from Bianca Pichler, Martin Stefanec, Asya Ilgun, Lilo, Valerin Stokanic, Martina Szopek, Stefan Schönwetter and Sarah Schönwetter and are fondly remembered and appreciated.

I am thankful for the many friends I have been able to make outside the university over the past four years. My roommates, Paddy and July, have been a constant source of support, good food and much needed motivation in times of dire need. I am greatly thankful for friends at the “ÖSM” and “International Fellowship” who made many a difficult weeks bearable and

sometimes even enjoyable. For the sake of brevity, I will refrain from naming all of these friends who I feel have been key players to my sanity and happiness in Graz.

My parents and siblings have proved time and time again that if all else fails in this world, I will not be alone facing whatever I face. I am exceedingly grateful and often amazed by their sacrificial support which is one of the few things I cannot imagine living without.

I would also like to thank the Knerich family who took me in and treated me like a son during my initial time in Europe. I remember with gratitude their thoughtful and well placed gifts such as my bicycle and workstation which served me on a daily basis.

At the core, I am a Christian. Contrary to the recent popular notion that true science requires an agnostic or an atheistic stance, I have found no good reason to deny my belief. I increasingly realize that I stand beside very many scientists of varying statures when I make such a statement. Without any further explanation, I would like to thank God for the peace, the joy and the assurance that I constantly experience through the message of the bible.

A special word of acknowledgment to the three blue cups at the Artificial Life Lab which have been the vessels delivering the necessary caffeine to my body to help me snap out of sleepy mornings.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

"Swarm intelligence" (Beni and Wang, 1989) is a field of research that was born out of the interdisciplinary efforts towards understanding concepts such as self-organization, intelligence, collective behavior etc. It is broadly defined as the "emergent collective intelligence of a group of simple agents" (Beni and Wang, 1989). Over the years, swarm intelligence researchers have modeled, tested and validated several swarm intelligence behaviors, drawing inspiration from natural systems.

Although research in swarm intelligence has been well underway for decades, there has been a recent spike in research interest in favor of self-organized systems due to the proliferation of connected computational devices. Due to the cheap availability of computational, memory and sensory devices, the number of available computers and robotic systems has increased dramatically over the past few years (Koh and Magee, 2006). Such a proliferation of connected devices has posed challenges to the traditional centralized approach to controlling systems. Therefore, engineering and research communities have increasingly been pursuing decentralized approaches for controlling collectives (Kim and Follmer, 2017; Zahadat and Schmickl, 2016; Gubbi et al., 2013). Swarm intelligence and self-organization offer many such decentralized solutions (Dorigo et al., 2004) which take inspiration from self-organized collective behaviors found in natural systems.

Due to the unprecedented confluence of research interest and the availability of suitable platforms to test swarm intelligence algorithms (ePuck, 2009; Mondada et al., 2004; Riedo et al., 2013; Thenius et al., 2016), many independent algorithms have been developed for various applications. However, there are few efforts dedicated to finding an underlying design structure in order to unify the principle behind the functioning of various swarm behaviors. In this thesis, a design paradigm is presented which unifies a diverse range of collective behaviors.

1.1 Problem statement

As mentioned above, a number of problem specific swarm intelligent solutions have been presented by various researchers over the past few decades. However, seldom has a unification of multiple collective behaviors been attempted. Some fundamental collective behaviors for instance include task allocation, collective transport, collective decision making, synchronization, aggregation and flocking. Different methods have been suggested with varying degrees of success for each of these collective behaviors. Often, each of these solutions requires fundamentally different underlying designs at agent level. In this research, I[1] present a paradigm for swarm communication based on traveling waves found in nature. I call this paradigm the Wave Oriented Swarm Paradigm or "WOSP". Using this communication paradigm, a swarm of agents can exhibit a number of common collective behaviors using only local information. In addition to its versatility, the approach uses single-bit and therefore minimalistic agent-to-agent communication. Apart from presenting the unifying paradigm for common collective behaviors, I also combine these collective behaviors to form more complex swarm behaviors and thus, enable a swarm to autonomously perform complex collective behaviors. Furthermore, I will demonstrate the usability of the paradigm by validating it on robotic platforms which have minimalistic communication capabilities.

1.2 Research questions

The main contribution of this thesis is the novel swarm communication paradigm which enables the unification of common collective behaviors in swarm robotics. In the following, the specific questions addressed in this thesis are listed.

1. Has unification of individual swarm behaviors been attempted before?
2. Can a simple minimalistic communication paradigm be designed which is able to produce a diverse range of collective behaviors?

[1]Hereafter, all references to the author are made in first person. Although there are co-authors on all papers written in relation to this thesis, I am the first author and the major contributor to most of the papers this thesis draws from. A detailed delineation of contributions can be found in Appendix A.

3. Is a single-bit directional communication between agents and basic motoring enough to produce a rich variety of collective behaviors?
4. Can emergent taxis be accomplished using this minimalistic communication paradigm?
5. How resilient is the taxis behavior to signal loss in the agent-to-agent communication in comparison to an alternate taxis method?
6. Can the less resilient taxis method be improved using the suggested paradigm?
7. Can the paradigm be demonstrated on a robotic platform to demonstrate its basic functionality?

1.3 Outline

In order to make the reading easier, a brief outline of the overall structure of the thesis is provided below.

- *Chapter 1.* **Introduction:** In this chapter, a brief overview of the thesis, its motivation and its outline is provided. Without going into the details of the thesis, the general topic - swarm intelligence is introduced and the structure of the thesis is briefly outlined to enable readability.
- *Chapter 2.* **Background and Related Research:** In this chapter, I provide the backdrop to the research conducted in this thesis. I examine existing work in various individual behavioral niches in swarm intelligence and try to identify unifying design patterns previously employed by researchers.
- *Chapter 3.***WOSP:** This chapter contains the main research output of this thesis. In this chapter, a swarm design paradigm, which enables swarms to exhibit collective behaviors using minimalistic communication, is provided. The publications this chapter is based on and my specific contributions to them are provided in Appendix A.
- *Chapter 4.* **Collective Emergent Taxis:** An algorithm using an emergent mechanism for a swarm to ascend an environmental gradient using WOSP is introduced in this chapter. The gradient taxis capability of the swarm using this algorithm is also examined. The publications this chapter is based on and my specific contributions to them are provided in Appendix A.

- *Chapter 5.* **A Resilience Case Study:** Using taxis as a case, a study of resilience of the communication paradigm of WOSP is conducted. This study compares the ability of two taxis algorithms to reach a predefined goal in presence of agent-to-agent communication failures. Subsequently, the less resilient algorithm is improved by plugging in the WOSP communication paradigm. The publications this chapter is based on and my specific contributions to them are provided in Appendix A.
- *Chapter 6.* **WOSP for Event Detection:** In this chapter, WOSP has been used as a design paradigm to develop an algorithm to detect anoxia events in the lagoon of Venice using simple robots. This chapter also conducts a laboratory experiment to test the proposed solution to event detection using WOSP. The publications this chapter is based on and my specific contributions to them are provided in Appendix A.
- *Chapter 7.* **Conclusion & Future Work:** In this chapter, I conclude the thesis by summarizing the presented paradigm and the associated results. I also discuss some avenues which are worth being explored in the future.

2 Background and Related Work

As introduced in Chapter 1, this thesis is about a design paradigm, which unifies a diverse number of collective behaviors under a single underlying mechanism. In order to provide a background for the research conducted, we examine concepts like self-organization, collective behavior, swarm intelligence and swarm robotics etc. in this chapter.

2.1 Collective behavior in biological systems

Nature teems with various kinds of life forms with varying individual capabilities. Most of these individuals, ranging from relatively simple organisms like slime mold to complex animals like human beings, vary in their abilities to survive in their environment as an individual. Most of these life forms have been found to depend not only on individual capabilities but also on group level dynamics of some form to survive and reproduce. Moving around as a swarm of individuals permits hunting together, foraging more efficiently, sharing food or collaboratively defending against predators to increase their collective probability of survival and reproduction (Eberhart et al., 2001). Figure 2.1 shows three spectacular examples of collective behavior: flocking behavior of birds, shoaling of fish and aggregation in bees.

The individual probability of survival in fish is increased by moving in schools and collectively performing escape maneuvers when a predator is detected by one of the fish (Brock and Riffenburgh, 1960; Hall et al., 1986). In addition to collectively performing escape maneuvers, the increased probability of survival of an individual fish is explained by the "many eyes" (Roberts, 1996) hypothesis. The "many eyes" hypothesis states that moving together in a group increases the average foraging time of individual fish, as the burden of scanning the surroundings is distributed among all

individuals in the group. Since this distribution of responsibility leaves individual fish with more time to forage, the probability of survival of an individual fish increases.

(a) Free image by Monica Volpin from Pixabay

(b) Free image by Manfred Antranias Zimmer from Pixabay

(c) Image from Artificial Life Lab, Graz

Figure 2.1: The images show three spectacular examples of collective behavior found in biological systems. (a) A shoal of fish swimming in formation (b) A flock of birds flying in formation (c) A group of bees forming an aggregate.

Bees exhibit similar collective dynamics at various stages and aspects of their life span. Schmickl et al. (2008) found that newly hatched bees randomly walk around, stopping only when they meet other bees. The stopping times of bees are found to be proportional to the "comfort" of the local temper-

ature. This behavior emerges into an aggregation of bees at the locality of optimal temperature. The waggle dance of bees is another example of their dependence on collective behavior. The foraging efficiency of a hive increases dramatically when bees perform waggle dances (Seeley, 1992) to inform other bees about food sources in the vicinity.

The synchronized blinking of fireflies (Buck and Buck, 1966, 1968) and the aggregation of slime mold cells to form a slug (Devreotes, 1989; Siegert and Weijer, 1992) are other examples of life forms relying on collective abilities for foraging and reproduction. Various kinds of life forms with varying physical and cognitive complexity have evolved to perform decentralized behaviors in order to ensure greater probability of survival. The part each individual has to play in such behaviors is often simple, yet the resulting collective behavior that either adds up or emerges on a group level is often resilient to failures of individuals or other disruptive events (Middleton and Latty, 2016). Numerous natural swarms following simple behaviors of its individuals are scalable and can therefore consist of hundreds, thousands or even millions of individual entities (Kennedy and Wigglesworth, 1951; Simpson et al., 1999).

2.2 "Self-organization" to "Swarm robotics"

In the above section, I briefly described some examples in behavioral biology where scientists unraveled the simple rules behind the seemingly complex collective behaviors found in nature. Most of the collective activities of these biological systems are *self-organized* and are based on local interactions between individuals with limited information and influence.

2.2.1 Self-organization

According to Camazine et al. (2001) self-organization is defined as follows:

> *"Self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying*

> *interactions among the system's components are executed using only local information, without reference to the global pattern."*

The following are some typically observable properties of self-organization. Ever since Ross Ashby (1947) first mentioned self-organization, various researchers have come up with a much more comprehensive list of properties of self-organization observable at various scales. However, the following remain the central properties of self-organization. For the research conducted in this thesis, it is sufficient to consider these central properties of self-organization.

1. Self-organization is observed in systems consisting of several individual interacting parts. The number of interacting parts often consist of tens, hundreds, thousands or in some cases millions of individual entities.
2. The individual parts of a self-organizing system interact locally and have limited and local information and influence.
3. As a result of the local interactions between individuals, spatio-temporal patterns are formed at various scales of observation of the system. Such patterns are either a sum of parts of the individual interactions or an emergent phenomenon, where the pattern is more that the sum of its parts.

Self-organized systems typically rely on the following mechanisms to produce the emergent global patterns (Bonabeau et al., 1999):

1. Self-organized systems rely on positive feedback. Positive feedback or amplification is a process, in which a certain aspect, quantity or behavior is amplified actively by the individual parts of the system. The waggle dance of bees (Seeley, 1992) to recruit foragers follows an amplification pattern and so does the trail laying of ants, where existing trails are strengthened by the existence of pheromone trails, which have not yet evaporated (Hölldobler and Wilson, 1978).
2. Self-organized systems rely on negative feedback. Negative feedback mechanisms serve to counteract the positive feedback mechanisms. The evaporation of pheromone trails can be seen as a counterbalancing mechanism or a negative feedback loop which enables the optimization between the number of ants following a trail and the distance of the food source from the nest.

3. Self-organized systems tend to amplify fluctuations. The inter-individual variability and the resulting randomness is crucial to self-organizing structures. Using the same example of foraging, lost foragers finding new food sources and recruiting more foragers to these food sources is an example of this mechanism.
4. Self-organization relies on multiple interactions between the individual parts of the system.

Sumpter (2005) added some more mechanisms that are responsible for self-organization in biological systems. His additions are "leadership", "selfishness", responding to a stimuli beyond a threshold ("response thresholds") and "synchronization".

2.2.2 Swarm intelligence and swarm robotics

After the first mention of self-organization by Ross Ashby (1947), theories of self-organization were used to explain emergent patterns in physics and chemistry (Nicolis, 1977). Subsequently, the macroscopic properties of biological systems were also discovered. Drawing inspiration from self-organization in natural systems, Beni and Wang (1989) used the term "swarm" in the context of cellular robotic systems. Although Beni initially considered the usage of the word "swarm" a buzz word (Şahin, 2005), the term and its variants such as "swarm intelligence", "swarm optimization" or "swarm robotics" have evolved to represent a certain class of self-organizing systems in both technical and biological systems (Beni, 2005). Bonabeau et al. (1999) defined swarm intelligence as "the emergent collective intelligence of a group of simple agents". Overall, one could say that biological and technical systems that employ the mechanisms and exhibit the properties of self-organization can be considered swarm intelligent.

The simplicity and decentralized nature of the individual's behaviors producing collective phenomena has attracted interest from fields such as computer science, robotics and engineering disciplines seeking to inherit properties of simplicity, resilience and scalability in engineered systems. Because of the economic and practical in-feasibility of testing swarm intelligence algorithms directly on a large number of robotic platforms, much of the work that is done in the area of swarm intelligence research remains in simulation (Eberhart et al., 2001). Apart from algorithms that are designed

to self-organize a group of physical entities, several optimization algorithms and simulations of biological behavior have also been done which enables scientists to understand natural systems better.

Numerous algorithms (Rubenstein et al., 2013; Labella et al., 2006; Hayes et al., 2003; Zahadat et al., 2015) enabling simple agents to accomplish complex tasks such as source localization (Hayes et al., 2003), task allocation (Labella et al., 2006; Zahadat et al., 2015), collective mapping etc. have been developed. In Rubenstein et al. (2013) and Decugniere et al. (2008), swarm intelligent algorithms are proposed that enable robots with very limited individual abilities to transport large objects in a collective manner. In Hayes et al. (2003), a distributed algorithm for localizing the source of an odor in an environment was proposed and tested on a swarm of robots. Apart from enabling a group of simple robots to perform complex tasks, various algorithms and methods have been suggested to control a swarm of robots of varying sizes to perform specific actions and tasks in a coordinated manner such as arranging themselves in a particular shape or responding collectively to external cues or stimuli (Christensen et al., 2007; Le Goc et al., 2016; Rubenstein et al., 2014b). Several optimization algorithms relying on emergent behaviors have been proposed and successfully implemented (Schmickl et al., 2008; Kennedy and Eberhart, 1995; Yang, 2009). "Particle swarm optimization" (Kennedy and Eberhart, 1995) is an optimization algorithm, inspired by flocks of birds, which is being used for multi-objective optimization. BEECLUST (Schmickl et al., 2008) is an optima finding algorithm, inspired by a swarm of newly hatched bees, that can be used to find global optima.

2.3 Common swarm behaviors

In the above sections, I traced how research in self-organization has enabled scientists to understand patterns found in living and non-living natural systems. The possibility of building resilient and scalable system level behaviors from simple individual level behavior has led to the development of decentralized algorithms inspired by nature, where many parts of a system cooperate to accomplish a goal. Several such "swarm intelligence" algorithms have been proposed for various such collective goals to be accomplished. For example, Winfree (1967) discovered that synchronization is

a cooperative phenomenon and that led researchers into developing various models (Kuramoto, 1975; Mirollo and Strogatz, 1990; Izhikevich, 1999) for synchronization of a swarm. Flocking, aggregation, collective decision making are other such collective goals for which behavioral algorithms have been developed. In this section, I will explore such behavioral algorithms or "swarm behaviors" that have been developed to accomplish various collective goals. Since this thesis deals with presenting a unifying underlying design paradigm (as mentioned in Chapter 1), I will briefly examine a wide range of commonly observed swarm behaviors in the following sections.

2.3.1 Synchronization

Synchronization is a phenomenon commonly found in nature (Camazine et al., 2001). Fireflies lighting up rows of trees in South-East Asia are a spectacular example of synchronization. Crickets chirping in unison forms their acoustic counterpart. Swarm intelligence and swarm robotics benefits from temporal entrainment in general due to the need for coordination between the inherently distributed entities in the system. Therefore, synchronization has been widely explored by researchers working with distributed systems such as swarm robotic systems, sensor networks etc.

Although synchronization is a phenomenon commonly found in nature, it had not been until the 1960s that synchronization was understood as a collective phenomenon. Winfree (1967) discovered that synchronization is a cooperative phenomenon and Kuramoto (1975) improved Winfree's phase based model and introduced a universal form for weakly coupled, nearly identical limit-cycle oscillators. Ever since then researchers have been interested in cooperative synchronization and the original model has been improved. The models for synchronization of coupled oscillators can be broadly classified into two categories: The "integrate and fire" model (Mirollo and Strogatz, 1990) and the "pulse coupled oscillator" model (Izhikevich, 1999). In the integrate and fire approach, the individual oscillators are continuously affecting each other while in the latter approach, the individual oscillators are only affected by neighbors' pulses during short periods of pulses.

Werner-Allen et al. (2005) and Rubenstein et al. (2014a) presented an implementation of synchronization on robots based on a modified version

of the general synchronization model presented by Mirollo and Strogatz (1990). The implementations differ from the original model presented by Mirollo and Strogatz (1990) in such a manner that each node or agent responds to signals received from their neighbors. While the model from Mirollo and Strogatz (1990) requires each node to immediately react to incoming signals by modifying its own internal timer, Werner-Allen et al. (2005) requires the nodes to record firing events by neighbors and change their time period in the next cycle.

Wischmann et al. (2006) demonstrated a firefly like synchronization using a simple recurrent neural network. Wischmann et al. (2006) used the synchronization to switch between foraging and homing behaviors to achieve improved performance for the swarm as a whole. The authors also demonstrated that stable and robust synchronization can be achieved even with limited local robot-to-robot communication. Perez-Diaz et al. (2018) examined the correlation between the time taken for synchronization and the individual velocities of mobile agents moving randomly in an arena. The authors examined the questions both in simulation using virtual robots and in practice with real robots. The authors concluded that for low velocities and high velocities, the synchronization speed is sensitive to the velocity while for mid-range velocities, the synchronization speed remains relatively independent of the individual agent velocity.

2.3.2 Flocking

Flocking is a mechanism by which individuals in a swarm stay in the vicinity of each other without being dispersed into the environment. Being around members of the same species is crucial for reproduction and subsequently the survival of the species. The advantages of being able to stay and move together as a coherent swarm does not only include reproduction. Foraging together, hunting together, protection against predators are key benefits of the ability of a swarm. Therefore, flocking is evidently a fundamental swarm behavior found in nature. Staying together without being disconnected from the rest of the swarm is also interesting for robotic swarms. Researchers have modeled, tested, validated and implemented various flocking behaviors on robots.

The pioneering model of flocking was published by Craig Reynolds in 1987

while working in the field of computer graphics (Reynolds, 1987). Reynolds formulated that virtual birds or "boids" can flock, following simple rules given that they know the distance to their neighbors and the neighbor's velocity. The individual boid followed three simple rules: velocity matching, following the center of mass of the flock and collision avoidance. While attempting to decode the simple rules behind the schooling of fish, Huth and Wissel (1992) offered an influential model. The researchers demonstrated that individual fish follow simple rules of attraction, repulsion and alignment with their nearest neighbors which produces collective motion of a fish school. A mathematical model for flocking like behavior was suggested by Vicsek et al. (1995). The Vicsek model essentially formulated flocking behavior by relating the orientations of individual particles to the orientation of their neighbors.

Variations of the models presented by Reynolds (1987), Huth and Wissel (1992) and Vicsek et al. (1995) have been implemented in swarm robotic systems. The challenging aspect about implementing flocking based on the above models is that most of them require velocity matching of some sort which is not easily accomplished in robotics. Turgut et al. (2008) presented a swarm robotic implementation of Reynolds' boids by developing a virtual heading sensor. Ferrante et al. (2012) introduced flocking models that do not rely on velocity matching and therefore enabling an easier implementation on swarm robotic systems. In Ferrante et al. (2012), the model relies on computing attraction and repulsion forces based on the distance to neighbors and uses this information to manipulate forward velocity and angular velocity in order to achieve flocking. Wang et al. (2013) designed a distributed controller for flocking without explicitly measuring the velocity. However, the authors used position information of the neighboring agents to estimate the velocity in combination with attraction and repulsion potentials to achieve flocking.

Baldassarre et al. (2003) employed neural network evolution to develop controllers to exhibit flocking. In their simulation experiments, the robots employing the evolved controller were able to flock together and move towards a light target. Depending on how the robots moved with respect to each other the evolved controllers demonstrated different kinds of flocking. The researchers demonstrated that in addition to traditional methods for modeling flocking, artificial neural networks are an alternative.

2.3.3 Aggregation

Aggregation is a behavior by which a group of entities gather at a common place. A spectacular example of aggregation is that of the aggregation of *dictyostelium discoedium*, a species of slime mold. In the absence of food, slime mold aggregates in order to form a slug (Chisholm and Firtel, 2004) to perform collective foraging in the environment. Bonner (1949) and Durston (1973) examined the communication behavior of slime mold cells which despite its simplicity enables various kinds of self-organized flocking and aggregation behaviors throughout different phases of their life cycle (Chisholm and Firtel, 2004). Other examples of life forms forming aggregates are that of bacteria, cockroaches, bees, penguins (Camazine et al., 2001) etc.

Apart from being a useful feature in nature, aggregation is also useful in swarm robotics. For example, a swarm of robots with limited communication devices searching the environment for some items of interest will need to meet to share the knowledge collected (Varughese et al., 2018b).

The most common approach to modeling aggregation behavior is that of a finite state machine. In Garnier et al. (2008), the authors presented a faithful reproduction of aggregation behavior as that observed in cockroaches. The embodied robotic agents in Garnier et al. (2008) were "alice-bots" following a simple set of rules. The agents could either move randomly in the arena or stop for a particular period of time. The stopping time was proportional to the size of the aggregate of other stopped robots. This behavior quickly resulted in an emergent aggregate of robots.

In Schmickl et al. (2008), the authors presented BEECLUST, an algorithm modeling the behavior of newly hatched bees searching for optimal temperature. This algorithm has been proposed not only for simple aggregation but also with an additional component of a swarm aggregating at a global optima (temperature in case of newly hatched bees). This behavior can be modeled as agents which randomly move around and stop only when they are in the near vicinity of another agent. The stopping time is proportional to the optimality of the local temperature where the agent stopped. Using the BEECLUST algorithm, a group of agents can aggregate at the global optima in an arena.

Apart from using finite state machines based on probability and neighborhood size, researchers have also resorted to the evolutionary approach to

solve the problem of aggregation. Trianni et al. (2003) and Soysal and Şahin (2006) are two examples of aggregation behaviors produced by evolving weights of neural networks on an agent based level to reward an emergent macro level behavior.

2.3.4 Collective decision making

Centralized systems make decisions by collecting information from different parts of their system and subsequently have the central processor make a decision. However, for a system consisting of many individual actors collective decision making is a more complex task and also a crucial aspect for coherent behavior. Biological systems solve this problem in various ways (Camazine et al., 2001) such as following a leader or relying on mechanisms that produce emergent collective decisions. According to Trianni and Campo (2015), algorithms for collective decision making can be broadly classified into three main mechanisms. In the first mechanism, the swarm waits for one entity to have enough information to make a decision and then propagates that decision within the swarm. Organizational structures following this mechanism can be found in form of hierarchies within societies of various life forms (Rabb et al., 1967; Ahl and Allen, 1996). The second mechanism is called opinion averaging in which all individuals constantly adjust their own opinion based on their neighbors' opinions until the entire swarm eventually converges to one opinion. This mechanism for collective decision making in robot swarms can also be found in groups of animals which use it for effectively navigating as a collective (Simons, 2004; Codling et al., 2007). The third mechanism is based on the amplification of a particular opinion to produce a collective decision. In this mechanism, each individual randomly starts with an opinion and then changes its opinion to other opinions depending on how often they encounter the latter opinion. The amplification mechanism is also found within animals such as the pheromone trails selection in ants (Beckers et al., 1990) or the temperature based site selection of young bees (Szopek et al., 2013). The underlying mechanism of collective decision making of the algorithm presented in this paper relies on the amplification of the mostly held opinions within the swarm which is associated with the second category of mechanisms presented in Trianni and Campo (2015).

Depending on the application, the design of collective decision making

in swarms takes inspiration from one of the aforementioned mechanisms. Some swarms utilize leadership hierarchies or a few informed individuals to make swarm wide decisions. Couzin et al. (2005) modeled local rules governing flocking and how a few informed individuals influence the movement of the group. Walker et al. (2014) examined how dynamically chosen leaders can be used to coordinate robotic swarm dynamics either by using a leader-follower scheme or by enabling an external entity to interact with the swarm.

One example for the amplification mechanism used in swarm robotics is the implementation of the approach presented by Amé et al. (2006) on robots (Garnier et al., 2009). Here, robots imitate cockroaches by randomly moving in the arena and stopping intermittently with a stopping probability. The stopping probability is proportional to the number of other cockroaches which have stopped in the vicinity. With this simple mechanism, the aggregate can be viewed as the physical realization of the collective decision of the swarm.

Gutiérrez et al. (2010) used opinion averaging where the measured distance between nest and food sources are collected from robots in the swarm. The noisy opinions (distance measurements) are averaged to agree upon a foraging location which is closest to the nest.

2.3.5 Collective transport

The most commonly found and widely studied case of prey retrieval or collective transport is that of ants. Kube and Bonabeau (2000) investigated how cooperative object transport is achieved by ants. Upon finding an object to be carried to the nest, an individual ant attaches itself to the object and pushes. If there are enough ants pushing in the direction of the nest, the object moves towards the nest. If the object does not move in the direction of the nest the ant changes its orientation and tries to push again. In case the ant is still unsuccessful it detaches from the object, reattaches at a different point and tries again. The above-mentioned behavior inspired by ants is the most common bioinspired decentralized collective transport strategy implemented in swarm robotics. Broadly speaking, the problem of collective transport can be decomposed into collective decision making regarding the motion direction and the coordinated application of force to the object in

the desired direction. There are several variants of how these individual problems are solved by different swarm robotic systems. The direction or trajectory of motion results from a collective decision which is solved either decentrally or by letting a single robot lead the rest of the swarm. Force application on the object is solved by lifting/grasping (Groß et al., 2006) the object or pushing/caging (Fink et al., 2008; Kube and Bonabeau, 2000) the object.

In Campo et al. (2006), the authors presented a scenario where a group of four robots collectively transported an object to a goal. Although the robots had a noisy estimate of the goal direction, the collective could negotiate a common direction to move the object and eventually transport the object to the goal. The robots in the above paper used grippers to grip the object and therefore, an attachment is ensured between the object and the robots. Thus, the problem of coordinated application of force is substantially simplified compared to the solution presented by Kube and Bonabeau (2000). Due to the absence of grippers in Kube and Bonabeau (2000), however, the trajectory of movement is more defined while the robots spent more time re-positioning themselves due to loss of contact with the object.

Alkilabi et al. (2017) developed a methodology where a group of robots controlled by dynamic neural network synthesized using evolutionary computation techniques were used to perform a collective transportation task. The best evolved controllers in simulations were tested on epuck robots to transport objects of various shapes and sizes. Even though the robots could not sense the force feedback from the objects being transported, the evolved controllers were able to effectively transport the object to its destination by synchronizing pushing actions. Groß and Dorigo (2009) developed controllers using artificial evolution and obtained various strategies for moving objects of different shapes and sizes. The evolved strategies to move the object were: connect directly to the object, connect to each other and move in a coordinated manner, encircle the object and move in a coordinated manner.

2.4 "WOSP": a novel unification of collective behaviors?

The above section outlines the example of self-organization in biological systems and adaptations of those behaviors in swarm robotics. As hinted to previously, self-organization is not only found in biological systems but almost universally in nature. Among the various self-organizing systems, there exist many underlying similarities which can be observed at different spatio-temporal scales. Researchers have been asking various questions in order to better understand self-organization and consequently apply the understanding gained from one niche to other areas. While the rewards of a search for a single unifying theory of self-organization has been so far underwhelming, many incremental gains have been achieved which have enabled us to understand self-organizing systems better. While writing an outlook for the future with respect to self-organizing systems, Sumpter (2005) conducted a study of the state-of-the-art concerning self-organizing systems. Some existent unifying structures related to self-organization can be found in this study. Sumpter (2005) discussed the attempts of various researchers to unify self-organizing behaviors by introducing some "ultracalculus" that would allow us to predict what happens as a result of numerous complex interactions between individuals. Self-organized criticality (Bak et al., 1988), boolean networks (Kauffman, 1993), cellular automata (Wolfram, 1984) were individual attempts to introduce such an ultracalculus. Although each of these introductions helped to further the understanding of many biological and physical systems, none of them proved to be truly universal. In Sumpter (2005), the author added that there might not be one universal theory or an ultracalculus for self-organization. Due to the behavioral complexity of each system, Sumpter is of the opinion that each system has to be considered separately to develop behavioral algorithms.

Sumpter (2005) and Camazine et al. (2001) examined self-organizing behaviors in biological systems and point out from existing research that most of these behaviors have simple underlying mathematical bases. However, our understanding of self-organizing systems has not always been triggered by finding the mathematical bases first. While commenting on the methodology for understanding collective behavior, Sumpter (2005) suggested the following steps: *1)* observe and classify how individuals in a system interact with each other, *2)* build behavioral algorithms based on the observations,

3) build a mathematical model based on the behavioral algorithm. This observation helps to separate different stages of self-organization research and to appropriately place the research that is presented in this thesis. In this thesis, I am not looking for an underlying unifying mathematical basis or an ultracalculus behind self-organization or swarm intelligence. Our work is more concerned with finding whether an algorithmic or behavioral bases for common collective behaviors can be found.

It is also worth noting that there have not been many explicit attempts of defining a unifying structure which is probably due to the limiting nature of such an underlying design structure. Sumpter (2005) and Reid and Latty (2016) are two of the few papers that explicitly mention unification of swarm behaviors on a behavioral level. However, both of these papers are limited to the observation of similarities on a high level without actually suggesting a framework. Sumpter (2005) observed that although researchers collectively see vague similarities between different self-organizing behaviors, there is no universal theory or method so far that enables the exhibition of all or most of the collective behavior seen in biological systems. In Reid and Latty (2016), the authors raised the question whether the collectives of slime mold fundamentally differ from other animal collectives. Reid and Latty (2016) demonstrated from existing research that collectives of very simple organisms like slime mold exhibit all eight 'principles' of collective behavior outlined by Sumpter (2005).

In swarm robotics, there have been some attempts to unify the implementation of swarm behaviors at the algorithm design level by various researchers. Research areas like "Human Swarm Interfaces" (Le Goc et al., 2016) and "swarm control" have some elements of unification involved due to the fact that a swarm of agents has to be able to execute multiple behaviors.

In Le Goc et al. (2016), a classical approach to controlling a swarm was employed where users could interact with and control a swarm of certain robots, referred to as "zooids", using gestures. The authors achieved a responsive swarm using an external projector for tracking the robots' positions, assigning a goal position for each individual robot and then utilizing classical motion control strategies like Proportional-Integral-Derivative (PID) control to move the agents precisely from the start position to the goal position. While the programmer has control of the precise movements of the swarms' members, the presence of a higher organizational entity

is inherently necessary and thus depicts a classical example of top down control.

There is a substantial amount of work related to controlling groups of entities. In the pioneering work of Craig Reynolds, he introduced self-propelled particles known as boids (Reynolds, 1999) which exhibited self-organized flocking and collision avoidance. Reynolds' boids were able to mimic a flock of birds whose individuals followed simple behavioral rules. Due to its simplicity and decentralized structure it is applicable to large swarms. Its focus is on the generation of realistic behaviors as found in natural flocks and hence limited in its versatility.

In O'Keeffe et al. (2017) a concept of self-propelled particles with internal oscillators, or "swarmallators", was introduced. Attractive and repulsive forces were then used in relation to the relative phase shift of the oscillators for generating a range of collective behaviors. The internal processes and states of the swarms' entities substantially influenced and determined the interplay between individuals producing a small set of collective phenomena. Although swarmallators were able to display a range of rich spatio-temporal patterns, they do not unify commonly found collective behaviors.

In Nagpal (2002), the authors introduced an algorithm for the self assembly of identical agents on a surface into a predetermined global shape. Multiple gradients were developed by propagating messages starting from the agents at the edges in order to develop a relative positioning system among the agents. Subsequently, various shapes were generated by manipulating the behavior of agents with particular gradient values. More complex shapes were achieved by repeating the process of generating gradients and folding along the specific areas of interest. A variation of the aforementioned idea was used to assemble various shapes in a self-organized manner in a thousand robot swarm (Rubenstein et al., 2014b). In Abelson et al. (2000), programmable self assembly and other similar research done by various researchers were unified as amorphous computing.

Among existing work, the paradigm presented in this paper exhibits most parallel characteristics with the approaches presented in amorphous computing, however, significantly differs in several points. Instead of multi-bit signals encoding information, as used for communication in amorphous computing, the presented paradigm already produces rich behavioral diversity with single-bit communication. Another key difference is that the

presented paradigm refrains from using "seed" agents (Nagpal, 2006; Nagpal et al., 2003) or global knowledge regarding edges and vertices in case of programmable self assembly for origami generation. Instead, agents initiate communication with the rest of the swarm decentralized and randomly. In amorphous computing, languages such as Origami Shape Language (OSL) (Nagpal, 2002), Growing Point Language (GPL) (Coore, 2004) were utilized in order to enable a user to program a swarm. In this thesis, the proposed paradigm will be used to program a swarm to perform collective behaviors and additionally, a meta control scheme can be designed for the swarm to perform these behaviors autonomously.

Generally, it can be said that not many attempts can be found in literature that unify basic swarm behaviors with regards to its underlying structure. In contrast to existing approaches, the paradigm presented in this work allows a swarm to inherit rich self-organized collective behaviors which are commonly found in nature. Additionally, instead of complex messages or encoded signals solely one-bit communication suffices for the presented behaviors and ultimately allows the design of both top-down control interfaces and autonomous swarms. In Chapter 3, the paradigm and its inspiration is introduced, which forms the crux of the research presented in this thesis.

3 WOSP: Wave Oriented Swarm Paradigm

In this chapter, a Wave Oriented Swarm Paradigm (WOSP) enabling the control of swarms with minimalistic communication and yet allowing the emergence of diverse complex behaviors is presented. It is demonstrated that even a 1-bit communication channel between agents suffices for the design of a substantial set of behaviors in the domain of essential behaviors of a collective. As detailed in Appendix A, this chapter is based on the following publications:

- Thenius, R., Varughese, J. C., Moser, D., and Schmickl, T. (2018). WOSPP - a wave oriented swarm programming paradigm. *IFAC-PapersOnLine*, 51(2):379 – 384
- Varughese, J. C., Hornischer, H., Thenius, R., Zahadat, P., Wotawa, F., and Schmickl, T. (2018a). Controlling swarms: A programming paradigm with minimalistic communication. *CoRR*, abs/1804.04202

3.1 Inspiration

WOSP draws its inspiration from traveling waves and pulse coupled oscillators found in biological systems. In order to gain a better appreciation of the paradigm, I will briefly look into each of these concepts before introducing the wave based single-bit communication paradigm for swarms. This wave oriented communication paradigm, WOSP, forms the basis for the rest of the research presented in this thesis.

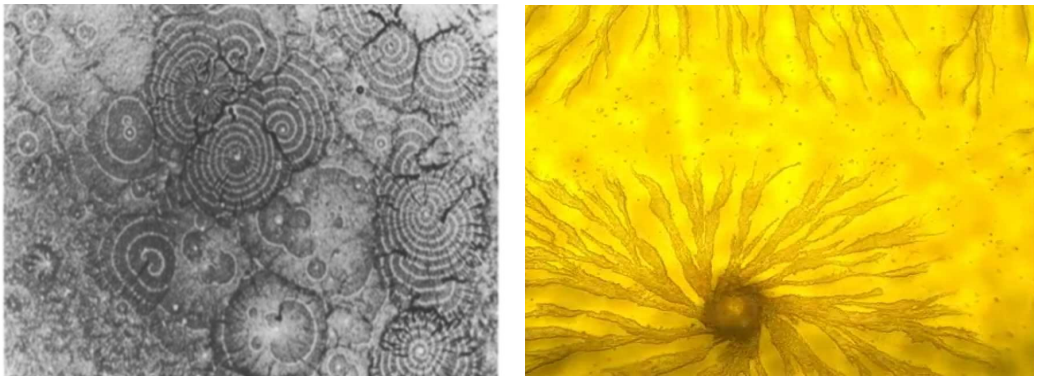
3.1.1 Traveling waves

The phenomenon of traveling waves in physical and chemical systems was easily observable and hence was studied early in scientific history. Traveling wave like oscillatory behavior in chemical reactions has been studied as early as in the 1600s by Robert Boyle. A spectacular example of traveling waves is that of concentration waves found in the autocatylic reaction of a bromate and malonic acid reagent, popularly known as the Belousov–Zhabotinsky reaction (Zaikin and Zhabotinsky, 1970). Similar wave patterns are found in various biological tissues like nerve fiber (Squire et al., 2012) and myocardium (Sherwood, 2015) where ionic potentials travel through nerve tissues or myocardium from one end of the tissue to the other. Calcium waves within frog eggs are another example of a biological system showcasing traveling wave phenomena (Bugrim et al., 2003). In the following section, I will take a closer look at the traveling waves produced during the aggregation phase of *dictyostelium discoideum*.

Scroll waves of slime mold

Slime mold (*dictyostelium discoideum*), is a free living diploid life form. It has been a subject of much study in the past due to its ability to survive harsh environments by taking advantage of group behavior. Each organism starts its life as a unicellular amoeba, but during starvation they aggregate to form a multicellular super organism. Chisholm and Firtel (2004) divide the slime mold life cycle as follows: aggregation, streaming, slug, culmination, fruiting body. In order to understand the traveling wave phenomenon, I will mainly look into the aggregation behavior of slime mold.

When there are ample food sources, cells grow and divide in a matter of three to four hours (Siegert and Weijer, 1992). On the other hand, if there is a scarcity of food, significant cooperation between the cells begins, hence, kicking off the aggregation phase. During this time, some cells (centers) release Cyclic Adenosine Monophosphate (cAMP) into the environment to induce a chemical concentration spike around them (Siegert and Weijer, 1992). cAMP concentration diffuses very quickly into the environment and therefore the chemical spike is short-lived. This chemical spike enables these centers to recruit other cells present around them. When surrounding cells perceive this chemical signal, they move towards areas of high cAMP



(a) Image from Durston (2013) (b) Image from Čejková (2015)

Figure 3.1: (a) Traveling waves of cAMP in *dictyostelium discoideum*. (b) A screenshot from a youtube video of *dictyostelium discoideum* aggregating into a slug.

concentration and release cAMP themselves, thereby relaying the signal. This, in turn, attracts other cells towards the centers. One cell is able to release cAMP at an interval of 12-15 seconds (Alcantara and Monk, 1974); during this interval, individual cells are insensitive to cAMP pulses. This interval can be understood as the refractory phase of the amoeba. The signal relaying mechanism described above forms the basis for spatio-temporal patterns known as scroll waves (Siegert and Weijer, 1992). The refractory phase is responsible for these scroll waves as it prevents the signaling organism from perceiving its own signal that was relayed earlier. The emergence of scroll waves enable the amoeba to move towards the recruiting centers for successful aggregation. The pictures in Figure 3.1 show the traveling wave phenomenon in *dictyostelium discoideum*.

3.1.2 Pulse coupled oscillators

Mutual synchronization occurs in many populations of biological oscillators. Some such synchronization of biological rhythms include networks of neurons in the circadian pacemaker (Winfree, 1967; Enright, 1980), pacemaker cells of the heart (Michaels et al., 1987), olfactory synchronization of menstrual periods of groups of women (Russell et al., 1980), acoustic synchronization of crickets chirping in unison (Walker, 1969), etc. Winfree (1967) discovered that synchronization is a cooperative phenomenon, which

triggered a great deal of research among physicists interested in nonlinear dynamics of many-body systems (Daido, 1988; Ermentrout, 1985). In this section, the spectacular synchronization of fireflies (*lampyridae*) is examined in detail.

Synchronization in fireflies

Fireflies (*lampyridae*) are a family of insects that are capable of producing bio-luminescence to attract a mate or a prey (Buck and Buck, 1966). The brightness of the bio-luminescent light depends on the amount of luciferin, a light emitting compound, available with the firefly (de Oliveira et al., 2011). Bio-luminescence of various families of fireflies has been a subject of elaborate study in the past (Buck and Buck, 1966). Apart from being able to blink, fireflies are known to behave in cooperation with other fireflies. It is spectacular to see thousands of fireflies light up in unison on a tree, lighting it up entirely. This uniform blinking happens so that the swarm has a higher chance of attracting mates or prey (Buck and Buck, 1966). The luminescence of the blinking swarm is much more than that of an individual firefly.

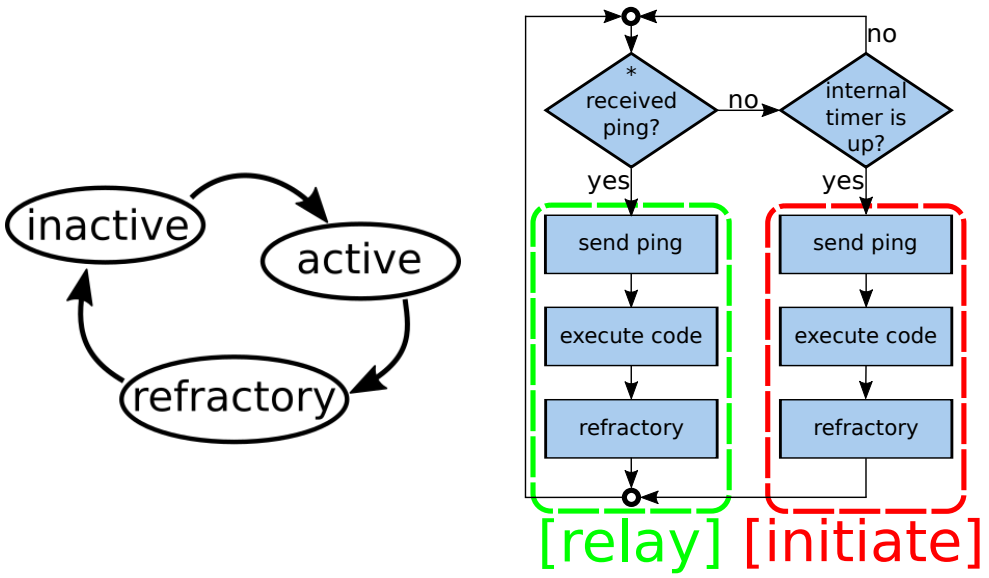
Such synchronicity is a result of a simple mechanism by which initially the individual firefly blinks randomly; when it perceives a blink in its surrounding, it blinks again and then resets its own frequency to match the other (Camazine et al., 2001). It takes time for the fireflies to achieve complete synchronization. This is analogous to a phase coupled oscillator which adjusts its phase to match it to that of the faster oscillator in the vicinity. This trait emerges into a pseudo synchronized blinking pattern while the frequency of blinking will be influenced by the fastest blinking insect. Figure 3.2 shows a photograph of fireflies lighting up in unison.

These existing models of slime mold and fireflies focus on the fidelity of the model with respect to the actual biological phenomenon. In contrast to the existing approaches discussed in the above section, I am more interested in the emergence of periodic scroll waves and its applicability to perform various collective behaviors in a swarm of robots. Therefore, I model periodic scroll waves as an agent based model where each agent represents a simple state machine combining the scroll waves of slime mold and the periodicity of fireflies to produce periodic scroll waves. The model is presented in detail in the following section.

Figure 3.2: A photograph of fireflies lighting up in unison. Image by Hristo Svinarov.

3.2 WOSP - Wave Oriented Swarm Paradigm

The wave oriented swarm paradigm WOSP is inspired by the two aforementioned organisms: slime mold and fireflies. In particular, communication within the paradigm is based on cAMP waves propagating through a swarm of slime mold. Every agent within the considered swarm has the ability to send and receive single-bit (1-bit) information signals, which are henceforth referred to as "pings". All agents can enter three different states: an "inactive" state, in which agents are receptive to incoming signals (responsive to cAMP or pings), an "active" state, where they send or relay a signal (release cAMP or send ping) and optionally perform an action, which is followed by a "refractory" state, where agents are temporarily insensitive to incoming signals. This is schematically shown in Figure 3.3(a). Additionally, every agent has an internal timer, which counts down constantly. When it runs out, the agent enters the active state where it broadcasts a ping, thus initiating a ping wave through the swarm. The ability of fireflies to adjust and reset their individual frequency of "blinking" is the inspiration for the concept of internal timers in this paradigm. For most primitives, this timer is reset right after running out, causing an agent to ping.



(a) (b)

Figure 3.3: (a): Three states of agents in WOSP: Through external trigger (for example incoming ping) or internal trigger (timer), the agent transitions from the inactive state into the active state, where it sends a ping and optionally performs an action. Afterwards, it enters the refractory state, being insensitive to incoming signals until transitioning into the inactive state again. (b): The conceptual operating structure of an individual agent. If an agent in the inactive state receives a ping, it relays the signal by entering the active state and sending a ping itself. Depending on the primitive, a certain piece of code will be executed and then the agent enters the refractory state. If an agent's internal time is up it will initiate a ping following the same structure, however, executing a different code (specific to the primitive) in the active state.

This operational structure results in a wave like propagation of signals throughout the swarm. Agents in the inactive state get triggered to relay a signal, while the refractory state prevents the system from continuously signaling and thus, flooding the system. In Figure 3.4, the propagation of waves is shown for a swarm of agents, each agent represented by a dot with the color denoting their state. The inactive state is denoted in black, the active state in red and the refractory state in green.

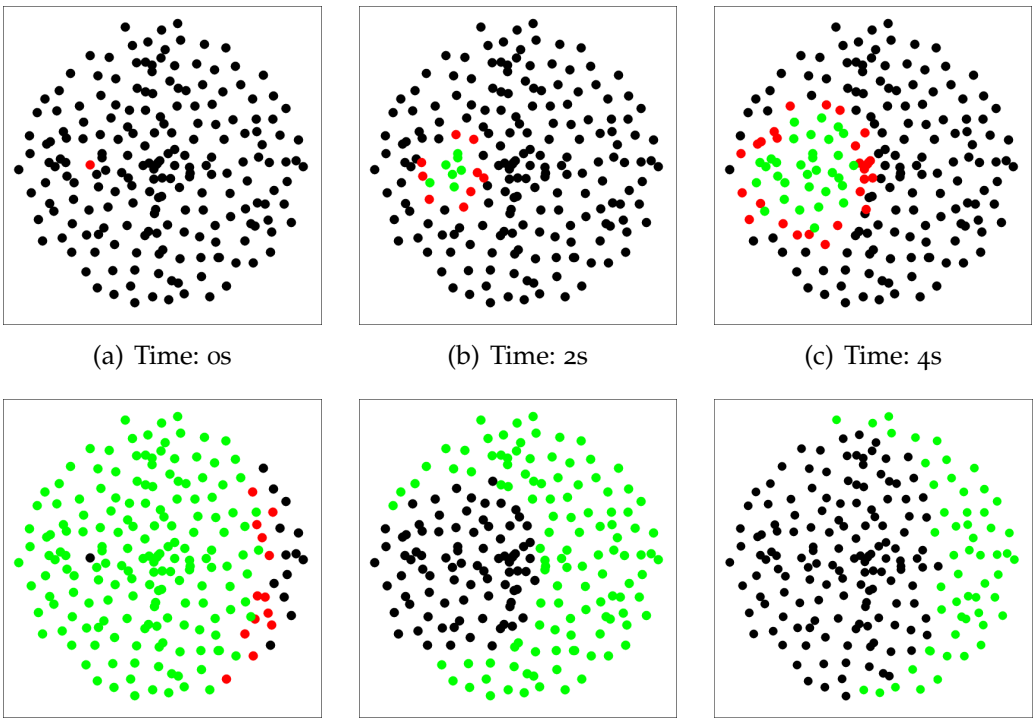
The aforementioned state transition constitutes the basic and fixed structure of an agent in WOSP and is shown as pseudo code in Algorithm 3.1. This suffices for the behavior shown in Figure 3.4 and is fixed for all agents. However, as will be presented later in this work, complex behavior can emerge when agents perform simple actions when relaying or initiating pings. This structure is conceptually shown in a flowchart in Figure 3.3(b).

For the behaviors or primitives presented in this work, agents are not only able to send and receive pings, but also have a heading and a sense of directionality for incoming pings. Furthermore, for some tasks, agents have the ability to move in direction of their heading. For demonstration of the idea of WOSP, the agents are considered as point particles and thus, collision detection is ignored in this work.

3.2.1 Parameters

The parameters and quantities used in this work are introduced and defined in this section. An analysis and discussion of the parameters is presented in Section 3.5.

- In the numerical simulations presented here, time is measured in timesteps s. An agent receiving a ping will relay the ping after one timestep $t_{activate} = 1\,s$.
- Every agent has an internal timer t_p which usually periodically resets to a maximum t_p^{max}. For some primitives, however, the timer can be reset to a random number between $t_p \in (0, t_p^{max}]$.
- The number of agents constituting a swarm is defined as N. The minimum N necessary for the presented primitives to function is $N = 2$. As further elaborated in the discussion, the maximum can theoretically be arbitrarily large, limited by the speed of agent-to-agent communication and operational timescales.



(a) Time: 0s (b) Time: 2s (c) Time: 4s

(d) Time: 11s (e) Time: 16s (f) Time: 19s

Figure 3.4: Illustration of wave based communication. In (a), almost all agents are in the inactive state, shown in black, except one agent which enters the active state and broadcasts a ping is shown in red. Afterwards it transitions into the refractory state, shown in green. Neighboring agents receive the signal and transition into the active state as shown in (b) and (c). The ping signal spreads in a wave like manner. In (d), the initiating agent transitions from the refractory state into the inactive state again. Due to a fixed duration of the refractory state, the transition into the inactive state as well spreads in a wave like manner, shown in (e) and (f). Parameters (as defined in more detail at the end of Section 3.2): number of agents $N = 80$, physical size of the swarm in units perception range $R_s = 5\,r$, refractory time in units timesteps $t_{ref} = 5\,s$.

```
Data: Paradigm parameters
Result: -
state ← inactive;
timer(t_p) ← random integer ∈ (0, t_p^max];
while primitive do
    decrement timer(t_p);
    if agent in refractory state then
        wait for refractory_time;
        if refractory_time is over then
            state ← inactive
    if agent in active state then
        broadcast ping;
        state ← refractory
    if agent in inactive state then
        listen for incoming pings;
        if ping received then
            state ← active;
            execute Relay-CodeBlock;
    if timer(t_p) ≤ o then
        state ← active;
        execute Initiate-CodeBlock;
Function Initiate-CodeBlock
    -
Function Relay-CodeBlock
    -
```

Algorithm 3.1: Basic pseudo code for every individual agent within WOSP. Behavioral changes are only introduced by adding commands to the initiate- and relay-codeBlocks which are highlighted here. The timer t_p is initially set to a random value with upper limit t_p^{max}.

- The refractory time t_{ref} denotes the time an agent remains in the refractory state, that is, insensitive to incoming pings. t_{ref} is set to a value as small as possible in order to maximize the time the agent is receptive to incoming pings, however needs to be set to a value sufficiently large so that a ping wave will not be relayed more than once.
- For distances the basic unit is the perception range of agents r, the distance up to which an agent perceives the pinging of a nearby agent.
- For primitives including locomotion, agents take discrete spatial steps within a timestep, where the length of their step d is set to one-sixth of a perception range $d = r/6$ if not stated otherwise. For decreasing d, the time until the tasks are completed increases, however, for too large values of d, the agents might move outside of the perception range with a single step and might thereby lose connectivity.
- The physical size of the swarm is defined as R_s and is given in units perception range r. If not stated otherwise, every agent in the swarm is initially randomly distributed within a circular area of radius R_s, such that every agent is connected to at least one neighbor.

3.3 Primitives

In this section, a set of primitives is presented, where small changes in the reactions of agents to incoming pings produce large scale complex behavior. For every primitive, both the plots of results as well as the code-block are presented. Furthermore, for each primitive presented, 20 independent simulation were successfully run if not stated or discussed otherwise. The presented primitives are divided into three categories:

1. **Internal organization** is about self-organization of the swarm on an internal level of each agent, including the primitives “leader election”, “synchronization” and “localize object”.
2. **Swarm awareness** includes the individuals’ awareness about properties of the swarm or properties of itself within the swarm. The primitives “localize swarm center”, “estimating number of swarm members” and “estimate extremities of the swarm” are presented.
3. The category **locomotion** is about physically self-organizing or restructuring the swarm, including the primitives “aggregation”, “moving

collectively" and "gas expansion".

3.3.1 Internal organization: leader election

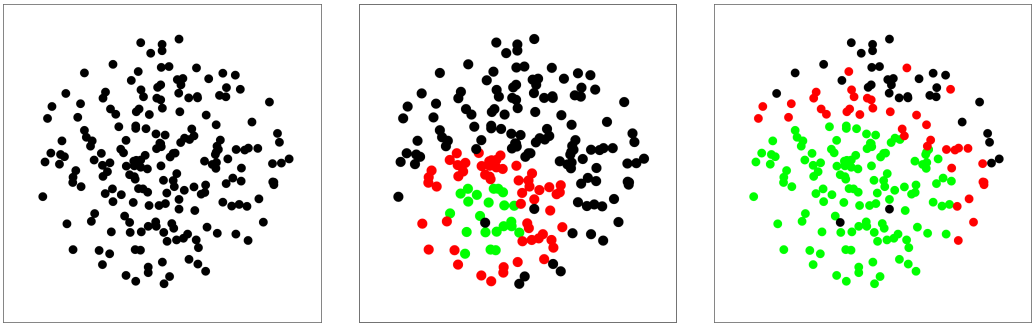
For various tasks it can be beneficial or even necessary for a swarm to have a certain agent "leading" a swarm. Having a certain agent assigned to a special entity introduces the risk of a single point of failure, thus, disabling the entire swarm. Instead, the swarm can collectively elect a leader, thus reducing such a risk. In order to choose a leader, all agents initially consider themselves potential leaders, shown in Figure 3.5(a) in black. An agent pinging is illustrated in red and an agent not considering itself a leader anymore is depicted in green. Every agent sets its timer to a random number within $t_p \in (0, t_p^{max}]$. As soon as an agent receives a ping before its own internal timer runs out it will not consider itself a candidate anymore and will also deactivate its internal timer. After an agent initiates a ping it will randomly choose a time t_p to initiate a ping again. This is shown as pseudo code in Algorithm 3.2.

Data: Paradigm parameters
Result: Leader election
.
.
Function *Initiate-CodeBlock*
  candidate ← true;
  timer(t_p) ← random integer $\in (0, t_p^{max}]$;
Function *Relay-CodeBlock*
  deactivate internal timer;
  candidate ← false;

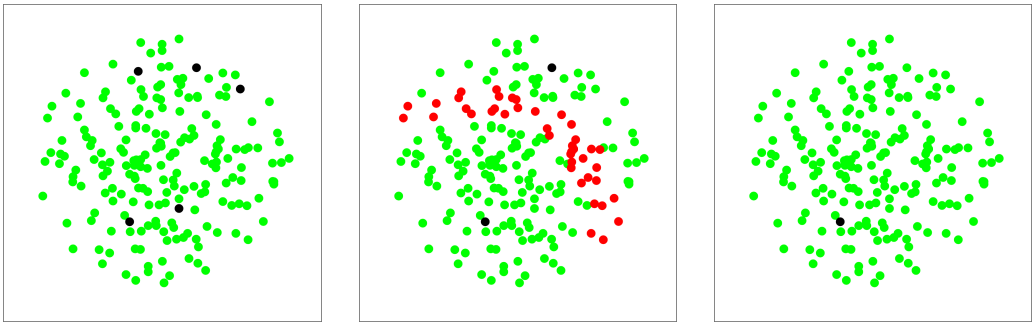
Algorithm 3.2: Code block for primitive "leader election"

Figures 3.5(b) and (c) show agents initiating ping waves and immediately outrivaling their surrounding agents. The refractory mode prevents two initiating agents from outrivaling each other, however, more than one can be left as potential leader, as shown in Figure 3.5(d). Since every remaining candidate again chooses a random time to ping, after few "negotiation cycles" a single candidate which then can be considered the leader, will

remain as shown in figures 3.5 (e) and (f). In order to empirically test the primitive, the primitive is said to have converged when there exists only a single candidate who considers itself the leader.



(a) Time: 0, t_p^{max} (b) Time: 0.005, t_p^{max} (c) Time: 0.007, t_p^{max}



(d) Time: 0.019, t_p^{max} (e) Time: 0.088, t_p^{max} (f) Time: 0.092, t_p^{max}

Figure 3.5: Leader election in a swarm. Candidate leaders are shown in black, pinging agents in red and agents that do not consider themselves a candidate for leadership are shown in green. Initially, all agents consider themselves potential leaders as shown in (a). After receiving and relaying a ping, an agent will not consider itself a candidate leader anymore. Agents initiating pinging thus outrival agents around them. In (b)-(d) it is illustrated how several agents initiate pinging and not outrival each other (due to refractory time). Since only candidate leaders will initiate pinging, the remaining candidates then repeat the process as indicated in (e), until only a single candidate remains, as shown in (f). Parameters: $N = 200$, $R_s = 3.34\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

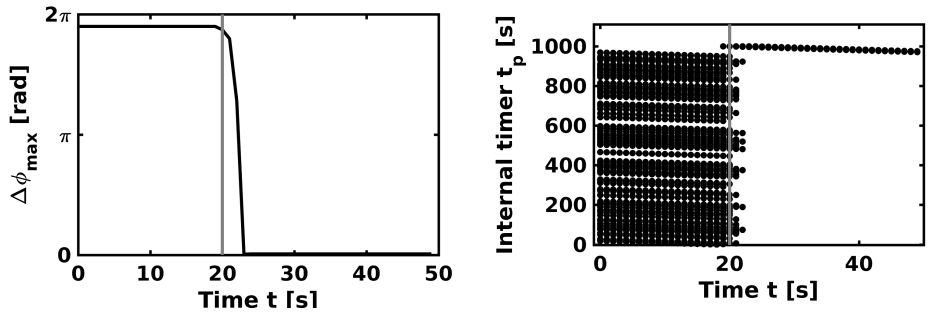
3.3.2 Internal organization: synchronization

It can be of great advantage for a swarm being able to perform coordinated actions. This primitive allows the swarm members to synchronize the sending of pings, allowing quasi-simultaneous coordination.

Every agent sets its internal counter to a random value between $t_p \in (0, t_p^{max}]$. If an agent receives a ping, it resets its internal counter to t_p^{max}. This is shown as pseudo code in Algorithm 3.3. As a result, the first agent sending a ping (which is then being relayed and propagates wave-like through the system) resets the timers of all relaying agents to the maximum t_p^{max}. Hence, the entire swarm will ping quasi-simultaneously within a time period smaller or equal to the duration of a ping propagating from one end of the swarm to the other. In Figure 3.6(a), the synchronization process for a swarm of $N = 50$ agents is shown via an order parameter $\Delta\phi_{max}$, which decreases with increasing synchronicity within the swarm. $\Delta\phi_{max}$ is calculated by determining the smallest phase interval containing the timers of all agents and then taking the maximum phase difference of all timer pairs. At $t = 30\,s$, the onset of synchronization is indicated with a gray vertical line. In Figure 3.6(b), the corresponding internal timers of all agents are shown, incrementally decreasing with time. Every line of points represents the internal timer of one agent. At $t = 12\,s$, an agent initiates pinging and thus resets the timers of all other agents such that at $t = 18\,s$, all agents reset and are thus synchronized. Synchronization primitive is said to have converged if $\Delta\phi_{max}$ is less than the phase difference corresponding to the time taken for a signal to propagate from one end of the swarm to the other.

Data: Paradigm parameters
Result: Synchronized Swarm
.
.
Function *Initiate-CodeBlock*
  timer(t_p)$\leftarrow t_p^{max}$;
Function *Relay-CodeBlock*
  timer(t_p)$\leftarrow t_p^{max}$;

Algorithm 3.3: Code block for primitive "synchronization".



(a) (b)

Figure 3.6: Analysis of the synchronization primitive. (a): Onset of synchronized internal timers at $t \approx 20$ of a swarm of $N = 100$ agents, indicated with a gray vertical line. The order parameter $\Delta\phi_{max}$ is plotted against simulation time t. $\Delta\phi_{max}$ is calculated by determining the smallest phase interval containing the timers of all agents and then taking the maximum phase difference between two timers. After fully synchronizing at $t \approx 20\,s$, the maximum phase difference decreased from $\Delta\phi_{max} \approx 5.5\,\text{rad}$ to $\Delta\phi_{max} \approx 0.04\,\text{rad}$, which corresponds to a time interval of $\Delta t \approx 5\,s$. This interval can be identified in (b), where the internal timers of the agents versus simulation time is shown. Every line of data points corresponds to the internal timer of a single agent, which incrementally counts down. All timers gradually decrease until at $t = 20\,s$ an agent's timer reaches $t_p = 0$ and thus initiates pinging. This marks the onset of the synchronization process and is marked with a gray vertical line. All agents relaying the ping then reset their timers. The reset signal propagates through the swarm and all agents reset and collectively count down in a quasi-synchronous manner. Parameters: $N = 100$, $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

3.3.3 Internal organization: localize object

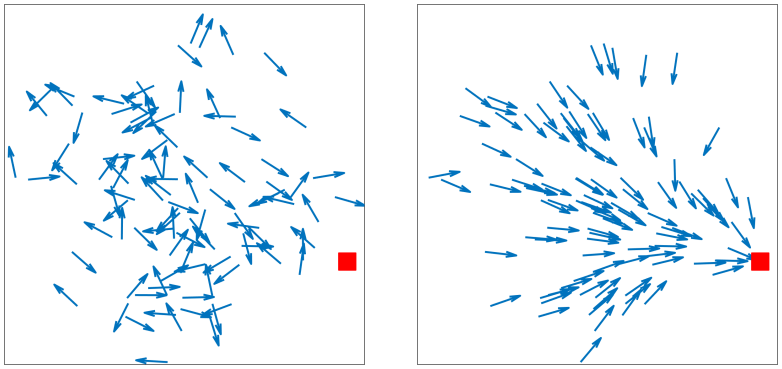
For distributing information about spatial structure of the surrounding, a swarm needs to be able to communicate the location of nearby objects or events among its members. This primitive enables a swarm to collectively localize the direction of a direct path towards an object which one or few members of the swarm detect. Each agent refrains from initiating a ping unless it detects an object. Every agent receiving a ping, records the direction of the incoming ping. An estimate of the direction towards the object is then obtained by taking a running average of the directions of incoming pings.

The pseudo code is shown in Algorithm 3.4.

Figure 3.7 shows the agents' estimate of the rough location of the object as an arrow placed at the position of the agent within the swarm. The red square represents an object which can only be detected by agents in its vicinity. Figure 3.7 (a) shows the initial (random) orientation of the agents. With an increasing number of perceived pings, the estimate of direction towards the object increases in accuracy until agents accurately point towards the position of the object as shown in Figure 3.7 (b). For the purpose of empirically testing the primitive, the convergence is said to be reached when the average error between the individual agents' perception of the direction of the object and the actual direction of the object is below a particular threshold.

Data: Paradigm parameters
Result: Agent knows rough direction of an object
.
.
Function *Initiate-CodeBlock*
    timer(t_p) $\leftarrow$ random integer $\in (0, t_p^{max}]$;
    if *no object detected* **then**
        state $\leftarrow$ inactive;
Function *Relay-CodeBlock*
    record ping direction;
    current estimate $\leftarrow$ average ping directions

Algorithm 3.4: Code block for primitive "localize object"



(a) Time: 0 t_p^{max} (b) Time: 0.49 t_p^{max}

Figure 3.7: The arrows representing the agents' estimation of the location of the object, the beginning of the arrow denotes the position of an agent. (a): The estimate is initialized with random direction at the start of the simulation. (b): The converged estimation of the location of the object after 0.49 t_p^{max}. All agents now point towards the object. Parameters: $N = 100$, $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

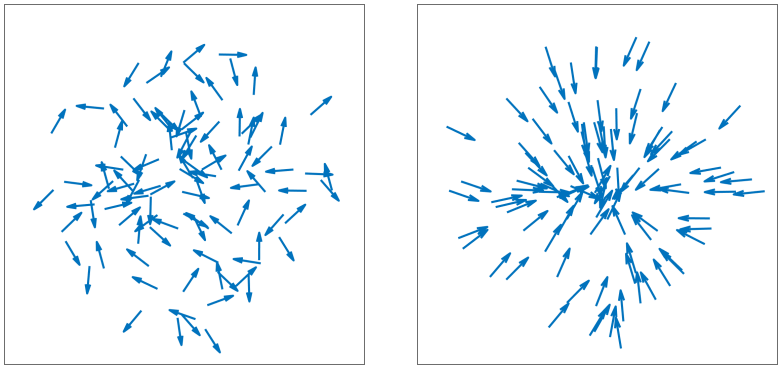
3.3.4 Swarm awareness: localize swarm center

For a swarm to be able to execute spatially coordinated actions, the knowledge of the individual about the location of the center of the swarm can be of great advantage. This primitive enables each swarm member to identify the direction from where most signals originate from, which will be referred to as Average Origin of Pings, or AOP. For a swarm of the presented type (circular, approximately homogeneously distributed, agents have several communication neighbors) this direction coincides with the direction towards the physical center of mass of the swarm. Each agent sets its internal counter t_p to a random value between $t_p \in (0, t_p^{max}]$ and as soon as a counter reaches $t_p = 0$ an agent sends a ping. When an agent receives a ping it stores the direction of the incoming ping and averages over all stored directions. This is shown as pseudo code in Algorithm 3.5.

Figure 3.8 shows a swarm in its initial state and after it equilibrated where every agent's orientation is denoted by an arrow at the position of the agent in the swarm. Initially, the heading is random. After equilibrating, the agents on the outside accurately point towards the center of the swarm.

Data: Paradigm parameters
Result: Agent knows rough direction of swarm center
.
.
Function *Initiate-CodeBlock*
    center estimate ← mean of previous estimates;
    timer(t_p) ← random integer ∈ $(0, t_p^{max}]$;
Function *Relay-CodeBlock*
    record ping direction;
    current estimate ← average ping directions

Algorithm 3.5: Code block for primitive "localize swarm center"

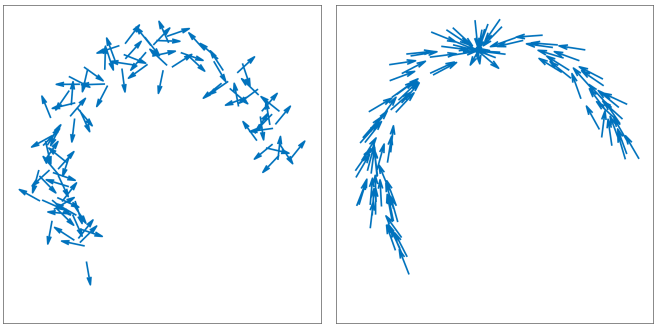


(a) Time: $0\,t_p^{max}$ (b) Time: $0.38\,t_p^{max}$

Figure 3.8: Agents' estimation of the direction towards the center of the swarm. The beginning of an arrow denotes the position of an agent. (a) shows the initial estimates of each agent as arrow at its position in the swarm. (b) shows the converged estimates after $t = 0.38\,t_p^{max}$. Parameters: $N = 100$, $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

However, this primitive does not result in the detection of the geometrical center of the swarm for all spatial configurations of agents since agents locate the direction from which they receive most pings. This is illustrated in Figure 3.9 where the swarm is shaped as a ring segment. Instead of agents pointing towards the center of the ring segment, they orient themselves towards neighboring agents, ultimately leading to the agents which would be in the center of the swarm if the swarm was stretched to a straight line.

When adequately selected, the t_p^{max} will be large enough for every agent to ping in a slot in which its ping wave does not collide with another wave, therefore providing a more accurate estimate of the average origin of pings. For empirical testing of the primitive, the primitive is considered to have converged when the average difference between the agents estimate of the direction of the center and the actual direction to the center is below a predefined threshold.



(a) Time: $0\,t_p^{max}$ (b) Time: $0.025\,t_p^{max}$

Figure 3.9: Agents' estimation of the direction towards the center of the swarm when it does not coincide with the geometrical center of the swarm. The beginning of an arrow denotes the position of an agent. (a) shows the initial estimates of each agent, (b) shows the converged estimates after $t = 0.025\,t_p^{max}$. Parameters: $N = 100$, radius of ring $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

3.3.5 Swarm awareness: estimating number of swarm members

For some tasks a swarm may need to be constituted of a certain number of agents in order to effectively operate. For instance, a swarm may need to check if the number of its members has substantially changed due to loss of members or merging with another swarm. The paradigm presented here enables the swarm members to decentrally estimate the number of swarm members without needing an external observer. Each agent sets its internal counter t_p to a random value between $t_p \in (0, t_p^{max}]$. Whenever an internal timer is up, an agent will initiate pinging and randomly reset its timer to $t_p \in (0, t_p^{max}]$. Each time an agent receives a ping it relays the signal and increments a counter N_{count}.

Furthermore, every time an agent initiates pinging (when one internal cycle has passed), it will store its counter N_{count} as its estimate of the number of swarm members for the past cycle. The average of those estimates will be the agent's opinion of the number of members in the swarm N_{est}. This is shown as pseudo code in Algorithm 3.6.

Data: Paradigm parameters
Result: Agent knows approximate number of members in the swarm
.
.
Function *Initiate-CodeBlock*
  estimate(N_{est}) $\leftarrow$ mean of previous swarm size counters;
  counter(N_{count}) $\leftarrow$ 0;
  timer(t_p) $\leftarrow$ random integer $\in (0, t_p^{max}]$;
Function *Relay-CodeBlock*
  increment counter N_{count};

Algorithm 3.6: Code block for primitive "estimating number of swarm members"

In Figure 3.10 the estimate N_{est} averaged over all members of a swarm versus simulation time is shown. The estimate quickly increases before slowly converging to $N_{est} \approx 34$. The error bars represent the standard deviation, thus indicating that the estimates of all agents are closely distributed around the mean. The estimate converges to a value significantly lower than the actual number of swarm members, however, for the same swarm the estimate consistently converges to the same (lower) estimate. The estimate converges to a lower value than the actual number of agents in the swarm because ping waves are initiated by more than one agent at roughly the same time, causing several ping waves to coincide. As a result, agents in the swarm detect only a single wave and accordingly increment the estimate only once. For larger cycle lengths, ping waves collide less and the estimate is closer to the actual number of agents. An empirical study on the dependence of N_{est} can be found in Section 3.5. To empirically test the primitive, all runs in which the estimate of the number of agents reached 75 % of N, that is $N_{est} \geq 0.75\, N$, were considered to be successful.

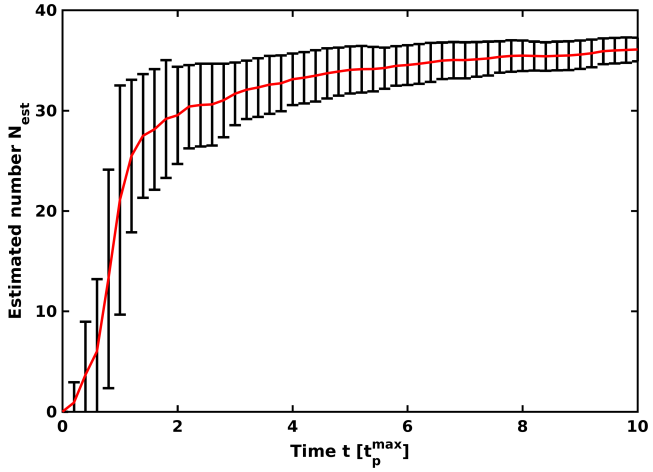


Figure 3.10: Estimated number of swarm members averaged over all agents in the swarm versus time. The error bars represent the standard deviation. The estimate steeply increases from $N_{est} = 0$ to $N_{est} = 30$ at $t = 3\,t_p^{max}$ before it gradually converges to its final estimate of $N_{est} \approx 35$. Parameters: $N = 50$, $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 2500\,s$. In order to put the timescales into perspective of cycle lengths, the time axis is measured as multiples of t_p^{max}.

3.3.6 Swarm awareness: estimate extremities of the swarm

This primitive enables the agents in a swarm to determine the extremities of the swarm by identifying gaps in the direction of incoming pings. In order to do so, each agent sets its internal timer t_p to a random value between $t_p \in (0, t_p^{max}]$. As previously explained, this will result in agents pinging at random time slots and each agent relaying the received pings. The agents then bin each of the pings received into four directions of $\alpha = 90°$ each. If there is at least one empty bin with no pings received, then the agent perceives itself as being on the periphery of the swarm. Pseudo code is shown in Algorithm 3.7.

Figure 3.11 shows the perception of agents regarding their position in the swarm. Initially, no agents perceive themselves to be at the periphery of the swarm, denoted by the black color of agents in Figure 3.11(a). As agents receive more pings from the surrounding agents, they are able to estimate their own position more accurately within the swarm as shown in Figure 3.11(b) where green colored agents perceive that they are at the

Data: Paradigm parameters
Result: Agent knows if it is at the periphery
.
.
Function *Initiate-CodeBlock*
  if *Is at least one bin empty?* **then**
    periphery ← true;
  else
    periphery ← false;
  timer(t_p) ← random integer ∈ $(0, t_p^{max}]$;
Function *Relay-CodeBlock*
  record ping direction;
  bin incoming ping directions into bins of 90°;

Algorithm 3.7: Code block for primitive "localize object"

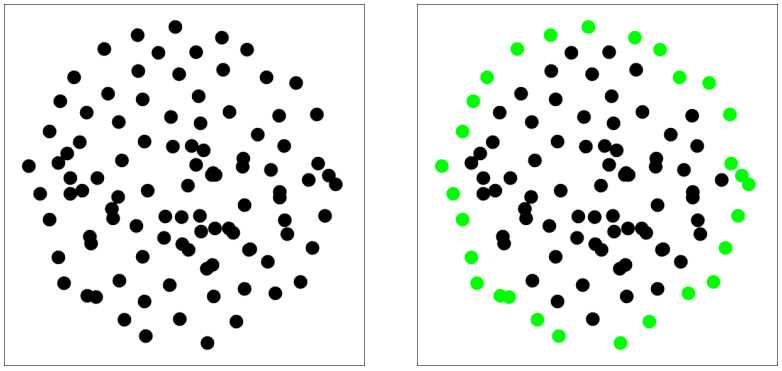
periphery of the swarm.

To empirically test the primitive, a simulation run was considered successful if more than 80% of the agents in the periphery successfully identified themselves as being at the periphery based on the incoming pings.

3.3.7 Locomotion: aggregation

Considering a swarm of agents with the ability to move and spatially arrange itself, it needs to be able to gather or aggregate in order to regroup itself. In order to achieve that, every agent randomly sets its internal counter t_p to a random value between $t_p \in (0, t_p^{max}]$. An agent receiving a ping will relay it and then move a small distance towards the incoming ping. Gradually, all agents move towards each other. The pseudo code for aggregation is shown in Algorithm 3.8.

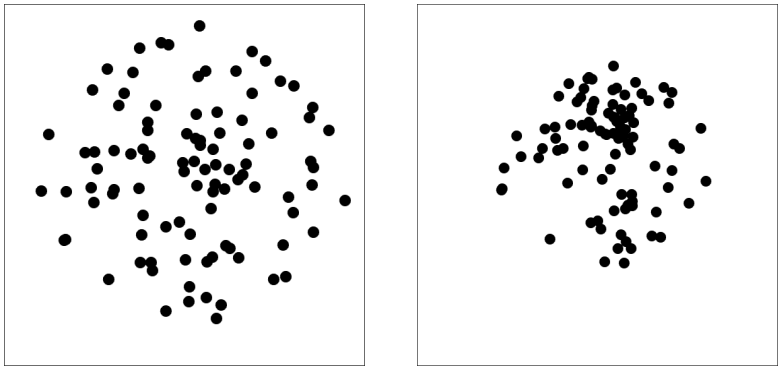
Figure 3.12 shows a swarm aggregating according to Algorithm 3.8. From its initial state the swarm steadily moves towards its average origin of pings, causing it to aggregate at the center of the swarm. Figure 3.12(d) shows the aggregated state of the agents as well as each agent's trajectory as blue line. This illustrates how agents tend to follow the paths of their fellow members of the swarm, producing a root-like trajectory structure. For illustrating



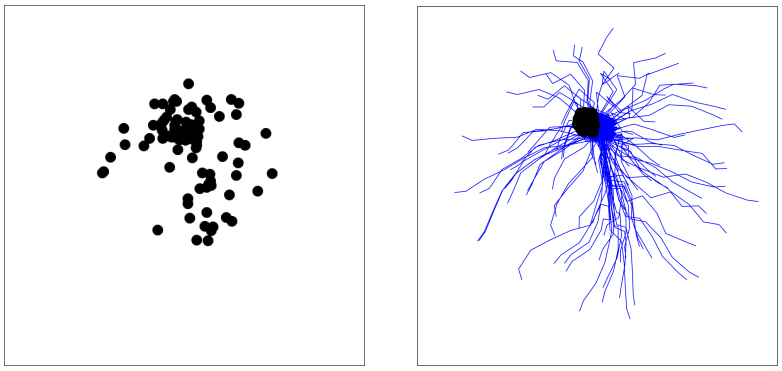
(a) Time: 0 t_p^{max} (b) Time: 1.4 t_p^{max}

Figure 3.11: Figures show the agents' perception of their location within the swarm. Green colored agents perceive that they are at the periphery and the black colored agents perceive themselves as not being at the periphery. (a) shows the initialization at the start of the simulation with all agents perceiving themselves as "not being at the periphery". (b) shows the converged perception of the agents after 1.4 t_p^{max}. Parameters: $N = 100$, $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

the aggregation process, Figure 3.13 shows in blue the average root mean square distance (R_{rms}) of all agents from the center of the mass of the swarm which represents the spread of the swarm members. To empirically test the primitive, a simulation run was considered successful with the termination condition $R_{rms} < 0.3r$.



(a) Time: $0\,t_p^{max}$ (b) Time: $0.075\,t_p^{max}$



(c) Time: $0.1\,t_p^{max}$ (d) Time: $0.35\,t_p^{max}$

Figure 3.12: Figures show aggregation of a swarm. Initial state of the swarm is shown in (a), in (b) and (c) it steadily aggregates. The final state and trajectories of each agent for the entire simulation are shown in (d) in blue lines. Parameters: $N = 100$, $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

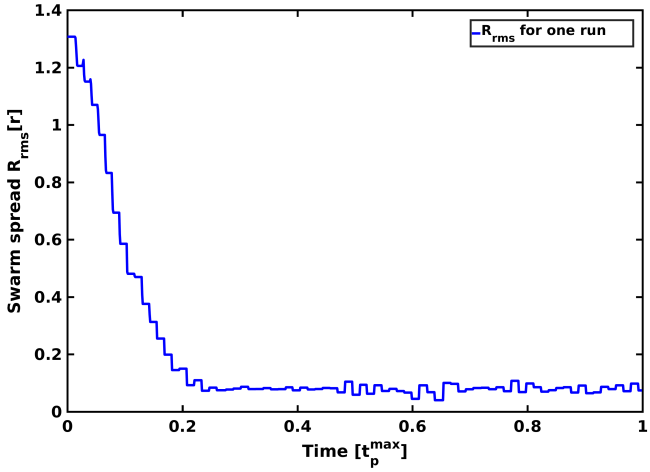
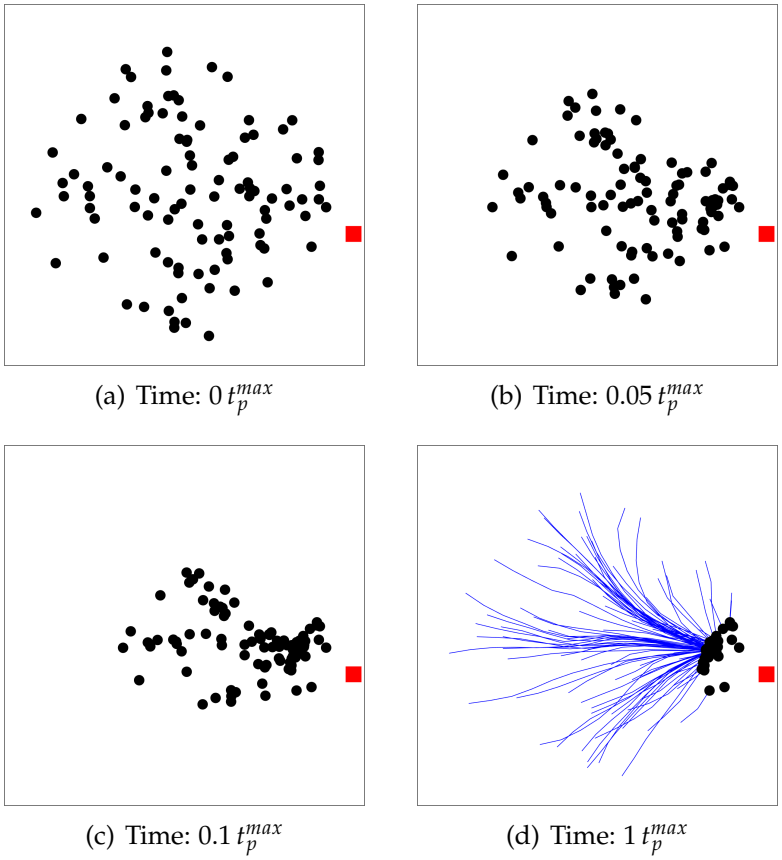


Figure 3.13: Average root mean square distance of all agents from the center of the swarm, R_{rms}, plotted against time. The blue line plot shows the R_{rms} of the swarm in the simulation shown in Figure 3.12. The R_{rms} linearly decreases until $t_p^{max} \approx 1$ when the swarm has almost fully aggregated. The decrease of R_{rms} occurs in discrete steps, corresponding to ping waves causing all agents to move towards each other quasi-simultaneously by a step of a predefined length d. Parameters: $N = 100$, $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

Data: Paradigm parameters
Result: Aggregated swarm
.
.
Function *Initiate-CodeBlock*
  timer(t_p) $\leftarrow$ random integer $\in (0, t_p^{max}]$;
Function *Relay-CodeBlock*
  timer(t_p) $\leftarrow t_p^{max}$;
  record ping direction;
  Calculate average of incoming pings;
  move towards incoming ping;

Algorithm 3.8: Code block for primitive "aggregation"



(a) Time: $0\,t_p^{max}$ (b) Time: $0.05\,t_p^{max}$

(c) Time: $0.1\,t_p^{max}$ (d) Time: $1\,t_p^{max}$

Figure 3.14: A swarm aggregating at an object, marked as red square on right hand side of the shown system. Initial state of the swarm is shown in (a). The swarm gradually aggregates at the object in (b) and (c) until every agent directly perceives the object in (d). The trajectories of agents for the entire simulation is shown in blue lines. Parameters: $N = 100$, $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

Considering that a swarm needs to aggregate at a specific location, the primitive can be changed in such a way that only certain agents, which for example perceive stimuli such as the presence of an object, are able to initiate pings. This is shown in Figure 3.14.

The stimulus can also be an event or can be connected with a gradient. Considering agents with the ability to perceive light intensity, the agents will be able to aggregate at the brightest spot if every agent sets its internal counter to a value proportional to its perceived brightness. The agents at

the brightest spots will statistically ping first. Furthermore, every agent receiving a ping will reset its counter, thus allowing the agents at the brightest spot to hijack the swarm. This process when repeatedly executed will result in a gradient taxis behavior as presented in Varughese et al. (2016) and in Chapter 4.

3.3.8 Locomotion: moving collectively

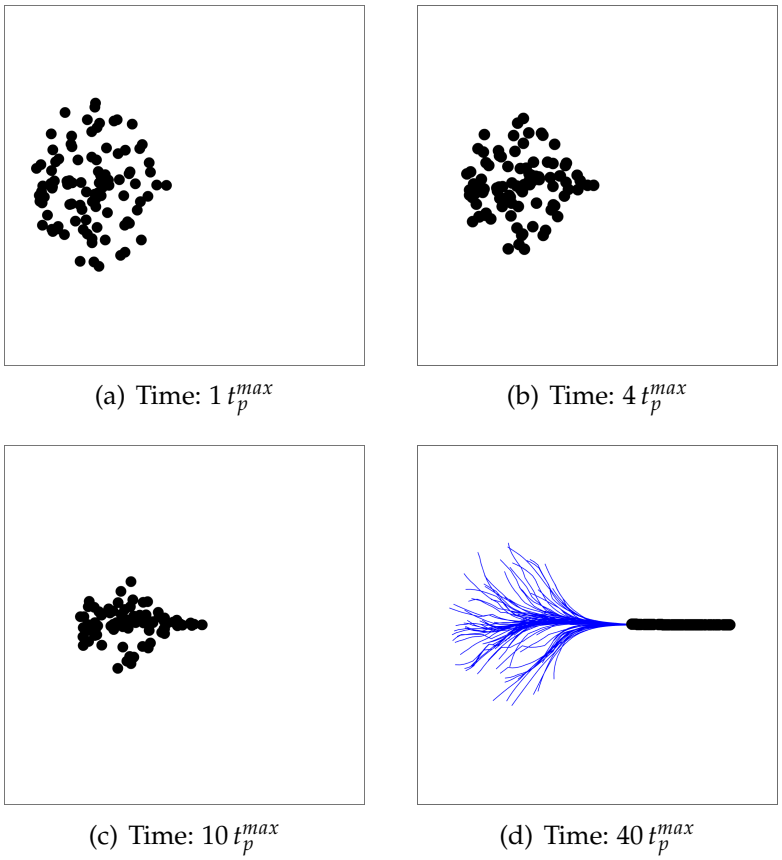
For the mobility of a swarm, the ability to collectively move to a specific location can be crucial. For letting the entire swarm move towards a certain direction, a single agent serves as leader. This leader exclusively initiates pings and gradually moves along a trajectory leading to the target location. All agents receiving pings will move towards the direction of it and thus, follow the leader. The pseudo code is shown in Algorithm 3.9.

Data: Paradigm parameters
Result: Swarm follows a leader
.
.
Function *Initiate-CodeBlock*
  leader $\leftarrow$ true;
  timer(t_p) $\leftarrow$ random integer $\in (0, t_p^{max}]$;
Function *Relay-CodeBlock*
  deactivate timer;
  leader $\leftarrow$ false;
  record ping direction;
  calculate average of incoming pings;
  move towards incoming ping;

Algorithm 3.9: Code block for primitive "moving collectively"

Figure 3.15 shows a swarm aggregating towards a leader located at the far right end of the swarm, which steadily moves towards the right. While following the leader, the remaining swarm forms a line behind it, being lead away. This primitive can be viewed as "aggregation at a specific, moving agent". For choosing a leading agent, the primitive "leader election", which was earlier introduced, can be executed prior to this primitive. This primitive

is said to have converged, when the R_{rms} of the swarm is below a set threshold, depending upon the size of the swarm.



(a) Time: $1\,t_p^{max}$ (b) Time: $4\,t_p^{max}$

(c) Time: $10\,t_p^{max}$ (d) Time: $40\,t_p^{max}$

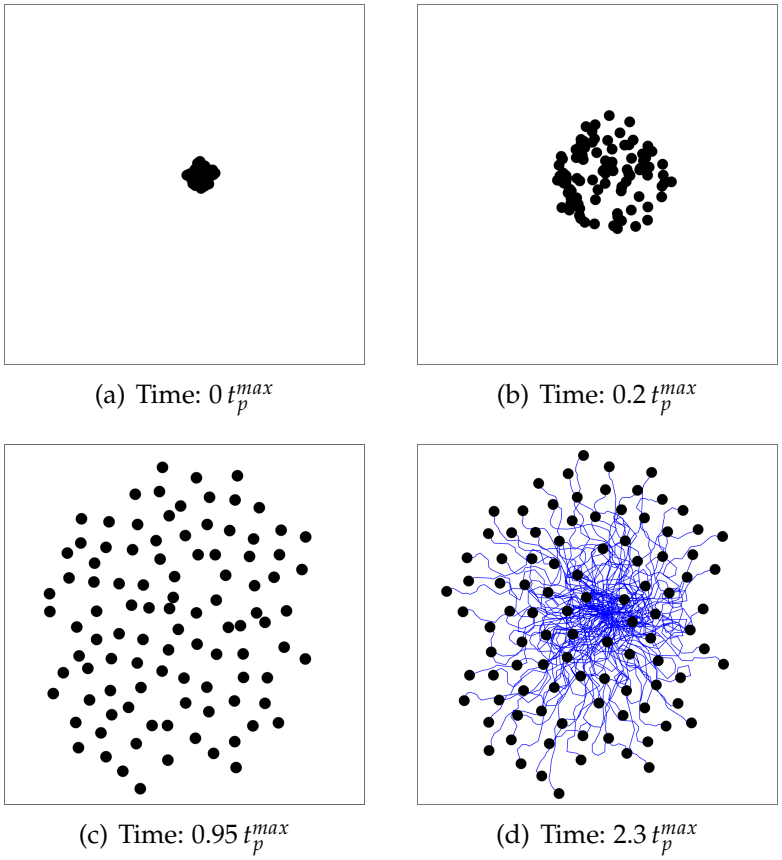
Figure 3.15: This figure shows a swarm being led by a single agent towards the right. From the initial state in (a), the swarm aggregates towards the leading agent in (b) and (c). In (d), the swarm forms a line following the leader. The trajectories of each agent for the entire simulation is shown in blue lines. Parameters: $N = 100$, $R_s = 2.33\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

3.3.9 Locomotion: gas expansion

The primitive "gas expansion" enables a swarm to uniformly expand. Each agent sets its internal counter t_p to a random value between $t_p \in (0, t_p^{max}]$. As soon as the internal counter reaches $t_p = 0$ an agent sends a ping. Each agent moves a small step away from the incoming pings. As soon as an agent does not receive pings anymore, it does not move any further. The agents can then reconnect with their swarm members by moving back, in the opposite direction of the previous step, or by integrating their entire trajectory and thus, finding their way back until they perceive signals again. Depending on the communication abilities of the swarm, the perception range or sensitivity can be temporarily decreased during the expansion such that afterwards the agents will again be connected. See Algorithm 3.10 for the pseudo code. In Figure 3.16 (a) an initially densely packed swarm is shown, which then expands in Figure 3.16 (b) and (c) until it is fully expanded in Figure 3.16 (d). To test the primitive empirically, a simulation run was considered successful if the average distance between any agent and its neighbors was greater than $0.8r$.

Data: Paradigm parameters
Result: Expanded swarm
.
.
Function *Initiate-CodeBlock*
  timer(t_p) $\leftarrow$ random integer $\in (0, t_p^{max}]$;
Function *Relay-CodeBlock*
  timer(t_p) $\leftarrow t_p^{max}$;
  record ping direction;
  calculate average of incoming pings;
  move away from incoming ping;

Algorithm 3.10: Code block for primitive "gas expansion"



(a) Time: 0 t_p^{max} (b) Time: 0.2 t_p^{max}

(c) Time: 0.95 t_p^{max} (d) Time: 2.3 t_p^{max}

Figure 3.16: A swarm performing the primitive "gas expansion". The initial state of the swarm is shown in (a), where it is aggregated. In (b) and (c) it gradually expands. The final state is shown in (d) with the trajectories of each swarm depicted as blue lines. Parameters: $N = 100$, $R_s = 0.67\,r$, $t_{ref} = 10\,s$, $t_p^{max} = 1000\,s$.

3.4 Combining primitives

By combining primitives more elaborate behaviors can be produced and by enabling a swarm to switch between a set of primitives it can operate autonomously. The most intuitive way of combining primitives is to execute primitives one after another in a sequential manner. This allows the design of complex tasks which can be executed by the swarm autonomously. This is schematically shown in Figure 3.17.

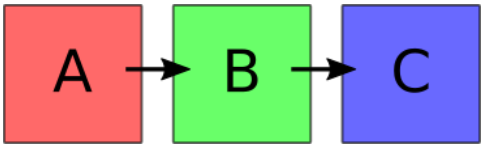
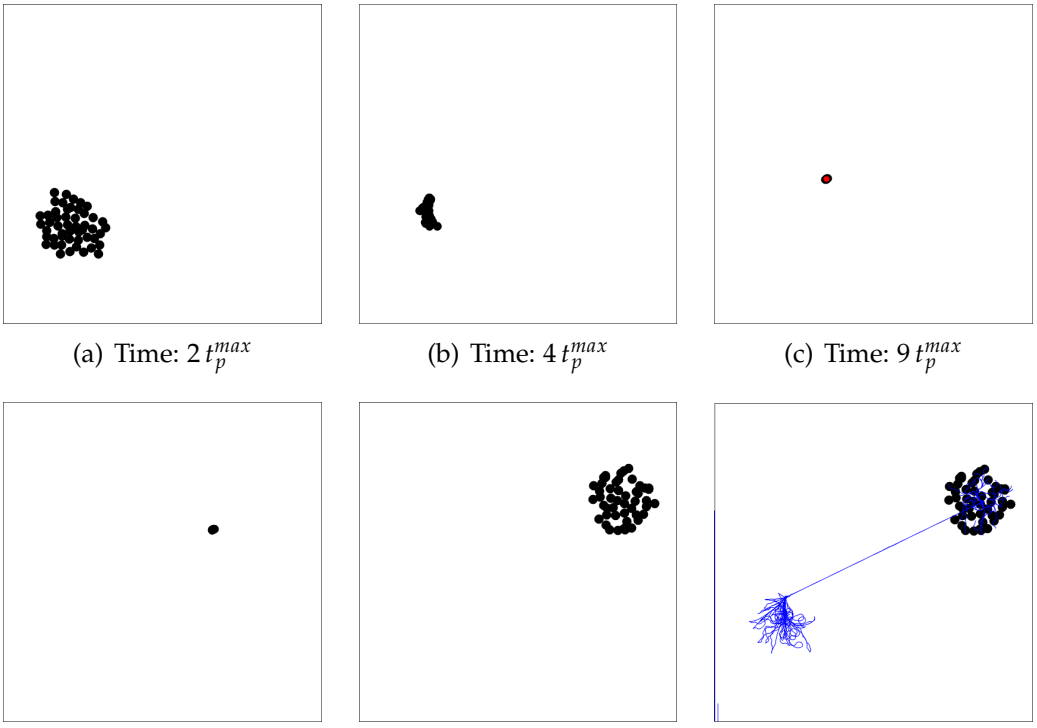


Figure 3.17: Schematic illustration of sequential execution of three primitives A, B and C.

3.4.1 Combining primitives: exploration

An example for sequential execution of primitives producing an autonomously acting swarm is a collective exploration procedure, shown in Figure 3.18. The following sequence of primitives is executed periodically: aggregation, leader election, moving collectively, gas expansion. In Figure 3.18(a)-(b) the swarm aggregates and then determines a leader in Figure 3.18(c)). This leader chooses a random direction and leads the swarm to a new location, as shown in Figure 3.18(c) to (d). Then, the entire swarm expands before aggregating again and restarting the same procedure. Due to the limited abilities of the individual members of the swarm, they have no awareness of the collective state or if the execution of a primitive has converged. For the present example, the execution times of all primitives were fixed.



(a) Time: 2 t_p^{max} (b) Time: 4 t_p^{max} (c) Time: 9 t_p^{max}

(d) Time: 10 t_p^{max} (e) Time: 13 t_p^{max} (f) Time: 14 t_p^{max}

Figure 3.18: Consecutive execution of the primitives aggregation, leader election, moving collectively and gas expansion as example for an exploring routine of an autonomous swarm. The swarm prepares for changing its location and thus aggregates from (a) to (c). It then decides upon a leading agent (marked in red) which then leads the swarm towards the top right of the system, a target area, shown in (c) and (d). The swarm expands again as shown in (e). In (f) the final state is shown along with the trajectories of all agents over the entire simulation. Parameters: $N = 50$, t_{ref} and t_p^{max} vary for each primitive.

3.4.2 Combining primitives: collective transport

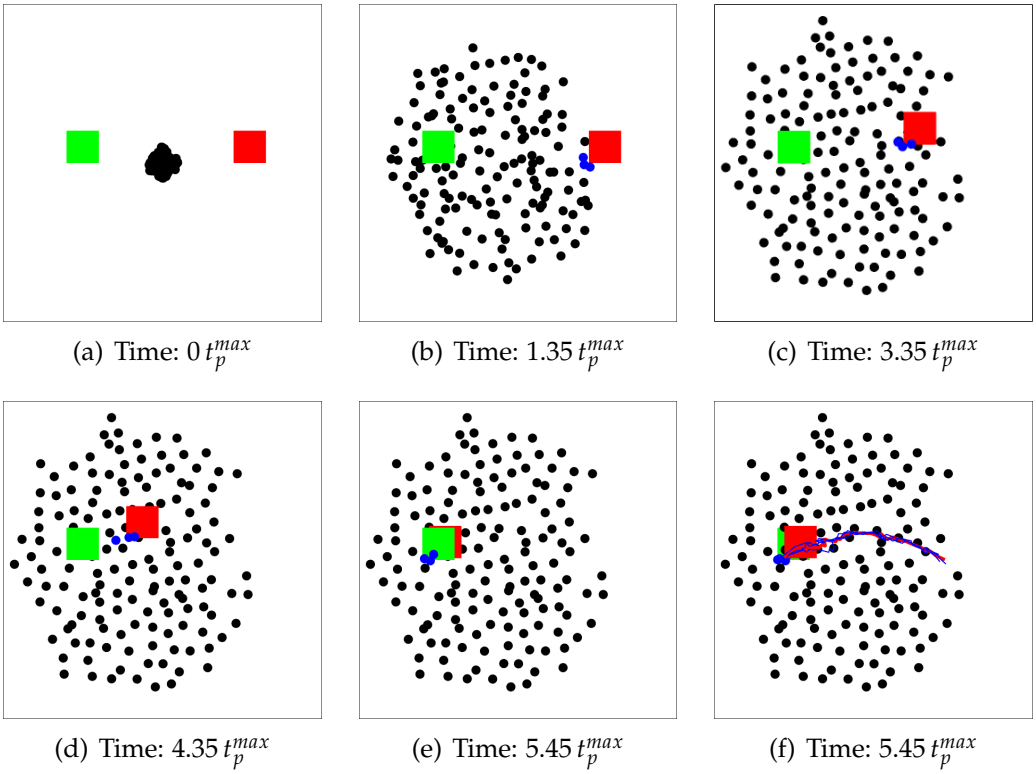
An instance of combining primitives in a sequential manner can be demonstrated by solving a simplified version of the collective transport problem. A scenario where a group of agents transports an object is presented here. In order to be able to move an object, additional physical abilities of agents are assumed. An agent can detect the presence of an object or a goal in

its near vicinity (one space unit). Additionally, it is also assumed that the agents can exert a force on the object. As for the motion of the object, it is assumed that if more than three agents exert force on the object, it can be moved along with the movement of the respective agents. The sequence of primitives executed here are as follows: gas expansion, aggregate at goal (only by agents which detected the object). The red and green patches in Figure 3.19 represent the object to be moved and the goal respectively. The swarm executes gas expansion in order to find both the goal and the object to be moved as shown in Figure 3.19(b). Those agents that detect the goal generate pings and those agents which find the object attach to the object and move it towards the incoming pings as shown in Figures 3.19(c) - (f). Through this combination of primitives, the agents can move the object to the goal without explicitly being told where the target is.

Evidently, the collective transport shown here is a simplified version. In order to present combining WOSP primitives to solve complex problems, complexities specifically associated with collective transport have been ignored. The area of collective transport has received much attention in the previous years. Broadly, the collective transport problem has been solved by either lifting/grasping (Groß et al., 2006) the object or pushing/caging (Fink et al., 2008; Kube and Bonabeau, 2000) the object. Using the approach presented here, the position of the goal is implicitly associate with the direction of the pings. Therefore, the need for each agent to know the position of the goal (global knowledge) is obviated. The advantage of using pings to guide the "pushers" to the goal comes at the cost of needing a large number of agents. The number of agents required for this task will depend on the sensor range of the agents. A large enough swarm with a specific sensor range can expand and discover both the source and the goal without losing connectivity.

It is important to note that the aim of presenting this scenario is not to present a perfect solution for the collective transport problem but rather to point out the wide range of possibilities using a minimalisitc communication paradigm. Using WOSP, a swarm programmer can combine several basic primitives with the physical abilities of the agents in order to accomplish complex behaviors.

Another approach to combining primitives is to execute several primitives in an interleaved manner. This allows the emergence of a larger variety of complex behaviors. For executing several primitives in a quasi-simultaneous



(a) Time: 0 t_p^{max} (b) Time: 1.35 t_p^{max} (c) Time: 3.35 t_p^{max}

(d) Time: 4.35 t_p^{max} (e) Time: 5.45 t_p^{max} (f) Time: 5.45 t_p^{max}

Figure 3.19: Collective transport by sequential execution of gas expansion and aggregation at the object. In (a), the red and the green colored patches represent the object and the goal respectively. In (b), the agents expand in all directions to find the object and the goal. Once both the object and the goal is found, the agents that found the object (shown in blue color) exert a force on the object to move it towards the incoming pings. Since the incoming pings originate from the agents that found the goal, the object is moved towards the goal as shown in figures (c) - (e). The trajectories of the object and the agents which move the object are shown in the red and the blue lines in (f).

manner, the previously presented single-bit communication can be extended. A simple option is to introduce several individual layers of single-bit communication, one for each primitive. Alternatively, multi-bit signals could be used to encode each waves of each primitive separately. However, interleaved execution of primitives is beyond the scope of this thesis and therefore, will not be presented.

3.5 Analysis and discussion of parameters

In this section, the influence of parameters and initial conditions of the swarm on the functionality of primitives are examined. Exemplary scenarios are chosen, which allow to infer the relation between parameters and the swarm's behavior.

3.5.1 Parameter dependencies

As seen in Section 3.3, the WOSP paradigm utilizes various characteristics of the incoming pings to infer global level properties of the swarm or to coordinate with each other to accomplish a common goal. For example, the estimation of swarm members is done by counting the number of pings in a single cycle length. The three main characteristics of incoming pings which contribute to the working of the primitives are: 1) the direction from which pings come in, 2) the number of pings received in one cycle length and 3) the timing of the incoming pings. Based on these characteristics, an analysis can be conducted and the dependence of the WOSP primitives on various parameters can be established. In this section, I will use "estimation of swarm members", "leader election" and "aggregation" to demonstrate the dependence of the performance of each of these primitives on their parameters. From such an analysis, inferences can be drawn on other primitives which employ similar characteristics of incoming ping waves.

Estimation of swarm members

The dependence of the estimate of swarm members on the t_p^{max} is shown in the color map in Figure 3.20, where the percentage deviation N_{err} from the

actual number of swarm members N depending on the maximum possible cycle length t_p^{max} is shown. For every data point the simulation was run for $25t_p^{max}$ which was sufficiently long for the estimate to converge. With increasing t_p^{max} the deviation from the actual number of members of the swarm decreases. Knowing the order of magnitude of N of a swarm, the t_p^{max} can be chosen to be sufficiently large such that the deviation is sufficiently small. For instance, considering a swarm of a maximum of 30 agents of the presented kind, a cycle length of $t_p^{max} > 1500$ would ensure a deviation as low as 10 %. It can be therefore said in general that for primitives that are dependent on the count of incoming pings, the performance of the primitive will depend heavily on the choice of t_p^{max}.

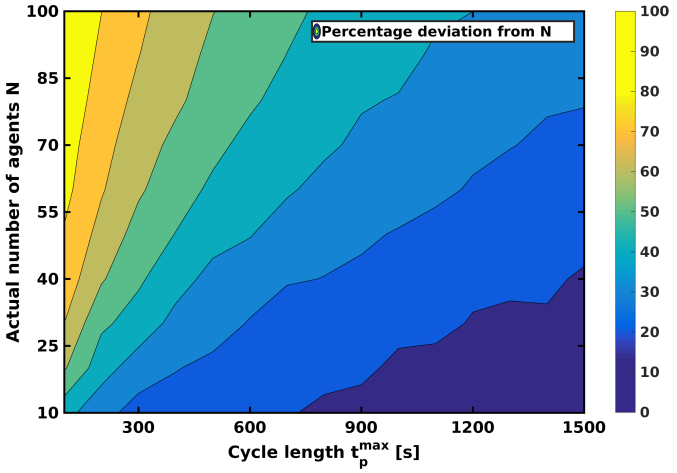


Figure 3.20: Percentage deviation of the estimated number of agents in the swarm versus cycle length and the actual number of agents in the swarm. Agents consistently underestimate the number of members of the swarm. The deviation decreases for decreasing N and increasing t_p^{max}. Parameters used: $N \in \{10, 20, .., 100\}$, $R_s = 1\,r$, $t_{ref} = 5\,s$, $t_p^{max} \in \{100, 200, .., 1500s\}$.

Time taken to elect a leader

An analysis on time taken for the convergence (τ) of the primitive "leader election" is performed with varying refractory time t_{ref} and cycle length t_p^{max}. Figure 3.21 shows that for a given number of agents, the selection of cycle length t_p^{max} affects the time taken to select a single leader. The refractory

time t_{ref} does not seem to have a significant effect on the time taken to elect a leader. For each data point, 10 simulation runs were conducted. Since the leader election primitive is dependent on the timing of incoming pings, similar conclusions can be drawn about primitives such as synchronization where timing of the incoming ping is crucial.

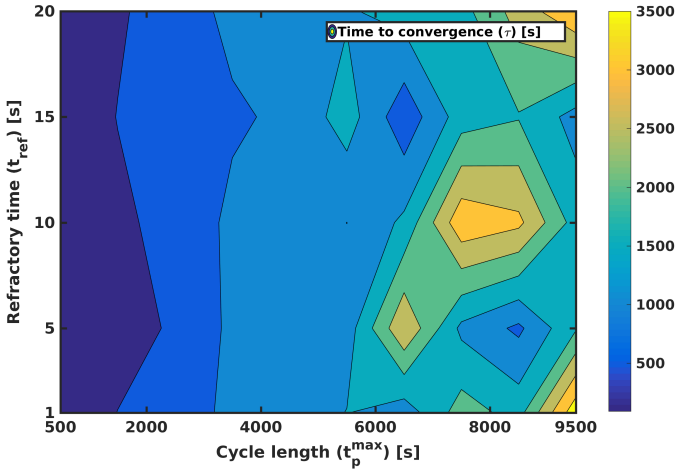


Figure 3.21: Time to convergence (selection of a single leader) is visualized, showing the effect of the refractory time t_{ref} and the cycle length t_p^{max}. As expected, the time to convergence (τ) rises with the increase in cycle length. t_{ref} does not seem to have a noticeable effect on τ. Parameters used: $N = 200$, $R_s = 3.33\,r$, $t_{ref} \in \{1, 5, 10, 15, 20s\}$, $t_p^{max} \in \{500, 1500, ..9500s\}$.

Aggregation success

Figure 3.22 shows the percentage success of the aggregation primitive being evaluated with varying lengths of steps d taken by agents and also the angular resolution $\Delta\omega$ of incoming ping direction. It can be inferred that at least an angular resolution of $\Delta\omega = 90°$ is necessary for consistent and successful operation of the aggregation primitive. Smaller step sizes allow for minimizing the effects of an inaccurate movement direction and therefore enable the primitive to be successful in aggregation more often. The dependency of aggregation on $\Delta\omega$ that is found here can be applied to other primitives which depend on the direction of the incoming ping. Therefore, it can be said that primitives such as “localizing swarm center”,

"localizing an object", "estimate extremities of the swarm" will also be similarly affected by $\Delta\omega$.

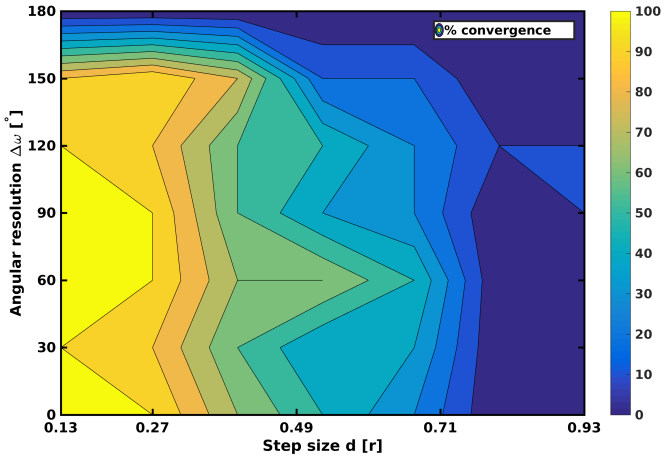


Figure 3.22: The percentage success of the aggregation primitive is evaluated with varying lengths of steps d taken by agents and also the angular resolution $\Delta\omega$ of incoming ping direction. It can be observed that when the length of steps taken by agents is below a third of sensor range, until about 90° angular resolution, the success of the aggregation primitive is not adversely affected. As the angular resolution decreases below 90°, the success of the primitive falls rapidly even for small step sizes. Generally, as the length of step d increases for any angular resolution, the success of the aggregation primitive suffers. Parameters used: $N = 100$, $R_s = 2.33\,r$, $\Delta\omega \in \{30, 60, .., 180^\circ\}$, $d \in \{\frac{0.4}{3}, \frac{0.8}{3}.., \frac{2.8}{3}r\}$.

3.5.2 Empirical analysis and choice of parameters

In order to empirically validate the observed collective behaviors of WOSP, each primitive was simulated 10 times with its respective convergence condition which is listed under each primitive. The results of this empirical simulation test can be found in Table 3.1. It can be observed that primitives reliably converge for the given set of parameters. This shows that a set of parameters can be found for the convergence of various primitives of WOSP.

When designing a swarm following this paradigm, parameters can be chosen according to the following guidelines. Using the time for an agent

Primitive	N	t_{ref} $[s]$	R_s $[r]$	t_p^{max} $[s]$	τ $[t_p^{max}]$	S [%]
Synchronization	100	10	2.33	1000	0.03	100
Leader Election	200	10	3.33	1000	0.06	100
Localize Object	100	10	2.33	1000	0.11	100
Locate Swarm Center	100	10	2.33	1000	0.23	100
Estimate Swarm Members	100	10	2.33	2500	6.17	100
Locate Swarm Extremities	100	10	2.33	1000	1.4	100
Aggregation	100	10	2.33	1000	0.14	100
Moving Collectively	100	10	2.33	1000	20.10	100
Gas expansion	100	10	0.67	1000	2.39	100

Table 3.1: The parameters which were used for the empirical analysis of the presented primitives are shown here. Apart from the parameters previously presented, τ refers to the average time taken for each of the primitives to converge and S is the convergence rate. A set of parameters can be reliably found for the convergence of each primitive.

to be activated by an incoming signal s as basic temporal unit and the perception range r as basic spatial unit, both usually determined by technical equipment, the speed with which a signal propagates is approximately $v_{ping} \approx r/s$. Estimating an upper limit for the spatial extension of a swarm R_s^{max} (considering its environment and its expected maximum number of N_{max} agents) the maximum time for a signal to propagate from one end of the swarm to the other is $t_{ee} \approx R_s^{max}/v_{ping}$. Once the the aforementioned quantities are determined for a swarm, the cycle length t_p^{max}, refractory time t_{ref} and step size of a moving agent, d,can be set accordingly.

Cycle length (t_p^{max})

For primitives where the accuracy with regards to counting the number of agents is used, the proper selection of t_p^{max} is crucial. t_p^{max} should be large enough to allow every agent to initiate a ping wave which should ideally reach all other agents in the swarm. In other words, each agent needs to be assigned a time slot (Q) in a cycle of length t_p^{max}, which is large enough before another agent pings, that is, the length of the $Q_l \geqslant t_{ee}$. In case the slots are assigned to individual agents in a top down manner, the

number of slots needed would be equal to the number of agents. However, if the agents pick a random slot to ping, there is a non-zero probability of ping collision which might affect the performance of certain primitives. Depending on the desired accuracy, the designer will need to choose the cycle length accordingly.

For a swarm of N agents, the probability that all agents select a different slot to ping, thus having no collisions between two pings, can be expressed as $P(no - collision)$ in Equations 3.1.

$$Slot\ length\ (Q_l) = \ max(t_{ref}\ ,\ t_{ee}) \tag{3.1}$$

$$No : of\ availableslots\ (Q_n) = \frac{t_p^{max}}{Q_l} \tag{3.2}$$

$$P(no - collision) = \frac{1}{Q_n} \cdot \frac{Q_n\ -\ 1}{Q_n} \ldots \frac{Q_n\ -\ (N-1)}{Q_n} \tag{3.3}$$

$$P(no - collision) = \frac{1}{(Q_n)^N} \prod_{i=0}^{N-1} (Q_n\ -\ i) \tag{3.4}$$

In order to avoid collisions in a self-organized system using the simple system of each agent picking a random slot, $P(no - collision)$ can be maximized according to the desired performance. The internal timer of each agent starts at a random initial condition in order to facilitate each agent choosing a random slot.

Refractory time (t_{ref})

The main consideration when selecting refractory time is the forward propagation of waves and avoiding the reactivation of the wave originator. Refractory time t_{ref} should be just large enough to avoid waves propagating through the system continuously and re-activating agents periodically. In Equation 3.5, k is the safety factor and needs to be appropriately chosen according to the application.

$$t_{ref}\ \geqslant\ k \cdot t_{ee} \tag{3.5}$$

Step size (d)

For the primitives in which the agents move, one main consideration is the distance moved by each agent per time step or the step size (d). Since connectivity is an important requirement for WOSP to function, maintaining connectivity is an important consideration. Therefore, the step size is selected in such a manner that the connectivity to the rest of the swarm is not lost due to large step sizes. Generally, the relation between step size and sensor range can be expressed as presented in Equation 3.6. In contrast to an agent based simulation for a realistic swarm, the sensor range is limited depending on the hardware that is employed, that is, r is a constant for such swarms. A safety factor L can then be suitably set so that connectivity in the swarm is maintained. In the simulations presented in this chapter, $r = 3$ space units and L is selected to be a sixth of r.

$$d \leqslant \frac{1}{L} \cdot r \tag{3.6}$$

3.6 Robotic experiments

In order to validate WOSP, robotic experiments were conducted with ten mobile robots capable of directional communication.

3.6.1 Setup

The robots were made by modifying the thymio-II robotic platform (Riedo et al., 2013) to be able to run the WOSP code and directionally transmit and receive signals. For this purpose, the thymio-II robots were integrated with a raspberry pi (Pi, 2017) and a blue-light communication module developed in Project subCULTron (Thenius et al., 2016). The raspberry pi was used to run the WOSP code and also to coordinate communications between the code, the blue-light board and the thymio-II robotic platform. The blue-light board consists of four blue light modules (a resulting angular resolution of $\Delta\omega = 90°$) which are able to send and receive 16-byte messages in their respective direction. The range of the blue-light modules used here is

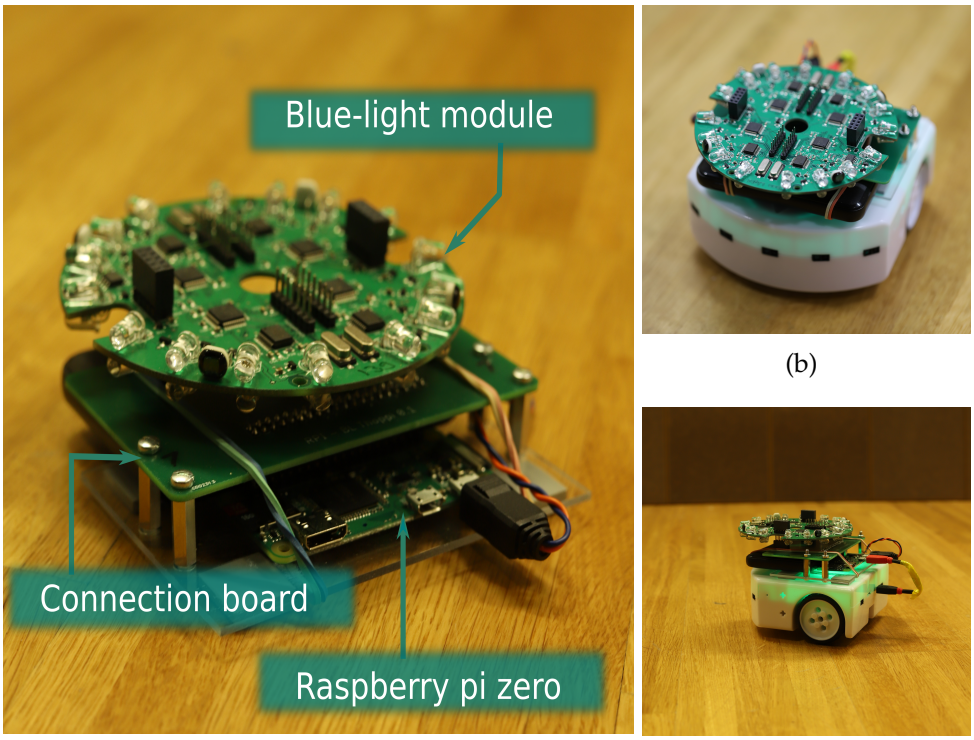


Figure 3.23: Photographs of electronic module integrated with thymio-II platform for directional communication. (a) Photograph shows the blue-light module which is integrated with raspberry pi zero to form the directional communication module. Figures (b) and (c) show the thymio-II robotic platform integrated with the electronics module.

about one meter (Thenius et al., 2016). The blue-light module was used to communicate using modulated blue light. Although the blue-light boards send 16-byte messages by default, the WOSP program developed here checked whether a message was received from the four different directions without considering the content of the received message in order to emulate single-bit signaling. The electronics module containing the connection board, blue-light module and the raspberry pi zero is shown in Figure 3.23(a). Figures 3.23(b) and (c) show the electronics module integrated on the thymio-II platform to enable the entire robot to move and communicate directionally.

Primitive	N	t_{ref} [sec]	t_p^{max} [sec]	d [r]	repetitions	S [%]
Estimate Swarm Members	10	10	300	0.1	10	100
Leader election	10	10	300	0.1	10	100
Aggregation	10	10	300	0.1	10	100

Table 3.2: Table showing the parameters used in the robotic experiments and also the empirical results of the ten repetitions conducted.

3.6.2 Experiments

The robotic platforms were programmed to execute the primitives, estimating number of swarm members, leader election and aggregation. The start of the experiment is shown in the photograph in Figure 3.24(a). Initially, all robots were randomly distributed in the arena and all agents considered themselves candidate leaders and therefore they have their green LEDs turned on. During the leader election primitive, all but one robot, are eliminated from leadership. All robots which were eliminated as leaders turned on their red LEDs as shown in Figure 3.24(b). Here, only one robot remains with its green LED turned on showing that it is the leader. Subsequently, all the robots aggregate at the leader as shown in Figure 3.24(c) during the aggregation primitive. Additional to the WOSP based aggregation found in the above sections, a simple obstacle avoidance behavior was also integrated into the aggregation primitive. The experiment was stopped when the thymio-II platforms were only executing obstacle avoidance behavior repeatedly. The trajectory of the aggregating robots executing the aggregation primitive is shown using a long exposure photograph in Figure 3.24(d). This sequential execution of primitives was repeated ten times to validate the experiment empirically. The estimate of population of the individual robots over ten runs are shown in Figure 3.25. It can be noted that the mean estimate of population for each robot (between 5 and 7) is below the actual value as expected. The deviation from the actual value of the number of swarm members is close to what was achieved in simulation as shown in Figure 3.10. As discussed under the respective primitive, the value of t_p^{max} can be adjusted for a better estimated of the total number of members in the swarm. All the parameters used for the experiment are shown in Table 3.2. The parameters, t_{ref}, t_p^{max} were selected according to the selection guidelines provided in the Section 3.5. The experiments were conducted on a flat arena.

A laptop computer was used to start the code quasi-simultaneously on all the robots.

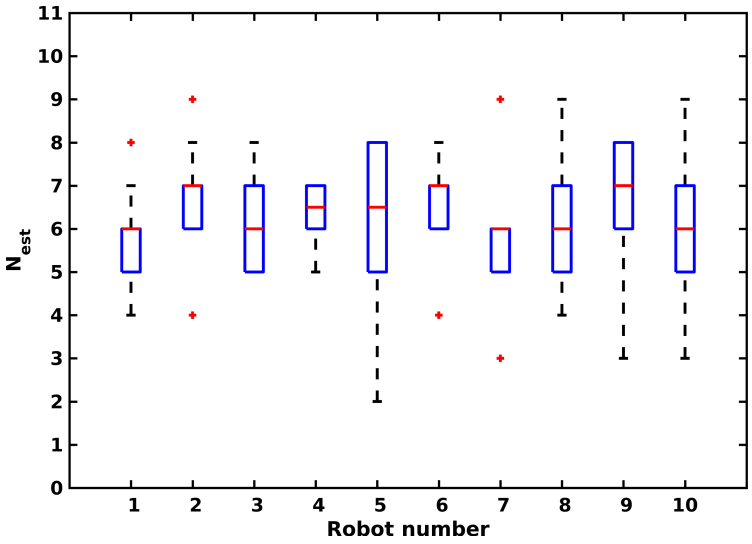
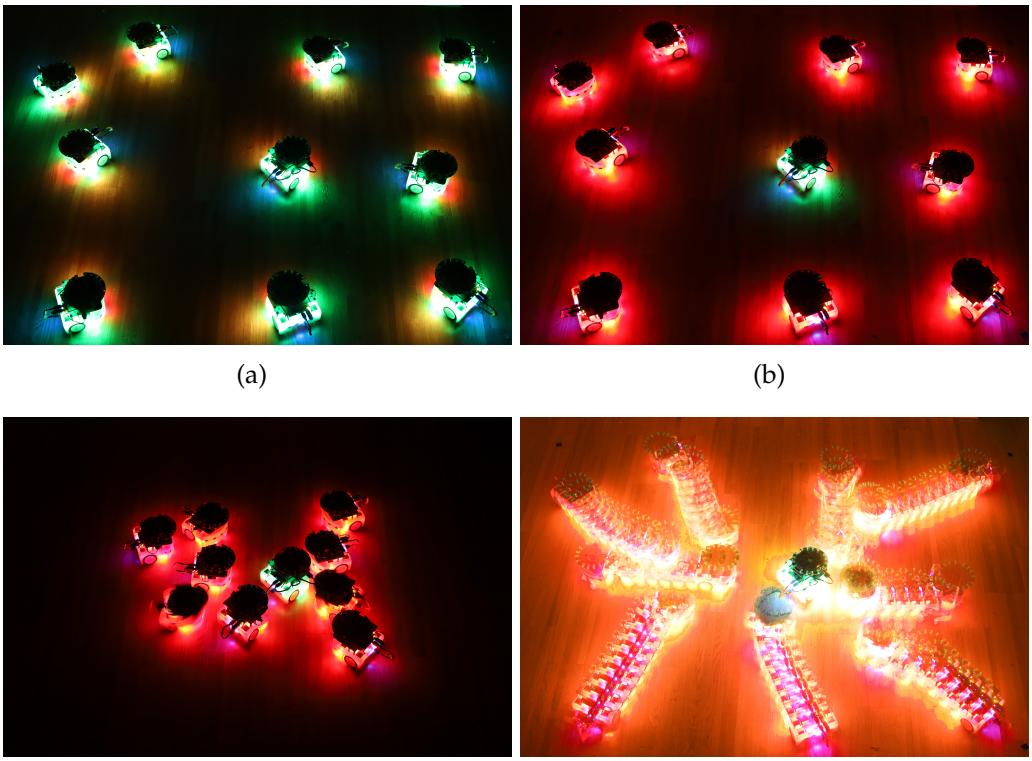


Figure 3.25: Plot showing estimate of population of each of the 10 robots in the swarm across 10 experiments. X axis shows the identity of robots $\in \{0, 1, 2 \ldots 10\}$. Parameters used: $N = 10$, $t_{ref} = 10$ *seconds*, $t_p^{max} = 300$ *seconds*.

3.7 Discussion

In Section 3.3, a set of primitives is presented, which can be utilized and combined as basic building blocks for a meta control scheme for a swarm, covering the categories "internal organization", "swarm awareness" and "locomotion". Two exemplary realizations of combination of previously presented primitives for complex collective behaviors are presented in Section 3.4. In this chapter, it is demonstrated that WOSP unifies common collective behaviors and thus enables swarms consisting of agents with limited abilities to collectively perform a large variety of complex behaviors. Due to its simple and flexible fundamental concept of "scroll wave" based communication, this paradigm is applicable to a large spectrum of different types of swarms and environments while requiring minimal communication abilities.



(a) (b)

(c) (d)

Figure 3.24: Photographs of various stages of WOSP validation using robotic experiments. An exemplary experiment is recorded in the following photographs. Figure (a) shows the start of an experiment. In (b), after the leader election primitive, one robot (shown in green) is elected as the leader. The non-leader robots have their red LEDs turned on. In (c), the final state of the experiment is shown where all the non-leader (red) robots have aggregated around the leader robot (green). In (d), a trace of the trajectories of the non-leader robots moving towards the leader while executing the aggregation primitive is shown through a long exposure photograph.

3.7.1 General features

The primitives of WOSP inherit a set of features due to the nature of the underlying communication paradigm. The features are discussed in the following sections.

Unification

Evidently, many primitives such as leader election (Karpov and Karpova, 2015; Ben-Shahar et al., 2014), synchronization (Perez-Diaz et al., 2018), aggregation (Trianni et al., 2003; Bahgeçi and Şahin, 2005; Schmickl et al., 2008), estimating the number of members in a swarm (Melhuish et al., 1999; Brambilla et al., 2009) are collective behaviors that have been implemented using various mechanisms. As shown in Section 3.3, WOSP unifies collective behaviors under a single minimialistic paradigm. Since the paradigm uses the same communication framework for all the primitives, the sequential combination of primitives can be easily accomplished as shown in Section 3.4. Apart from the conceptual simplicity of unifying behaviors, one benefit of such unification is that the same incoming signals can be used for inferring global properties of the swarm. For example, the number of agents in the swarm can be estimated at the same time as estimating the area of ping origin.

Minimalism

Minimalism is another benefit of WOSP that has been emphasized throughout the chapter. All primitives presented in Section 3.3 use single-bit communication. Although in the real world there is seldom such a stringent limitation of using such a narrow bandwidth to communicate between robots, the extreme assumption is made in order to demonstrate that much behavioral diversity is possible with a low bandwidth. In a real implementation of WOSP, a larger bandwidth will usually be available to the programmer. With an increased bandwidth, more primitives can be added and more easily combined with each other. In Section 3.4, a time based sequential execution of primitives is employed. A broader bandwidth will enable an easier parallel combination of behaviors. For example, if there are two bits of communication payload, one can be used to communicate

a detected obstacle and the other for the occasional pings from agents, the swarm can go around an obstacle and still stay together by moving towards the incoming ping from agents while moving at a certain angle to the incoming bit indicating the obstacle. In essence, more complex combinations can be done by increasing the bandwidth of communication.

Resilience

Resilience to ping loss is a benefit WOSP inherits from using scroll waves. An analysis has been conducted on the resilience of scroll wave based communication (Varughese et al., 2017) where its robustness against signal loss was examined. It was shown that due to redundancy in signal pathways, a system using slime mold based communication, as utilized in WOSP, can compensate up to 70% individual probability of signal loss without significant decrease in performance. The ability of this basic behavior to cope with high amounts of signal loss endows WOSP with resilient functioning when pings fail to be sent or received. A detailed analysis of this aspect can be found in Chapter 5.

Scalability

As opposed to approaches such as the one presented by Le Goc et al. (2016), decentralized control in WOSP allows scalability limited primarily by the communication abilities relative to operational time scales. For the class of swarms presented in this chapter the main constraint to scalability is constituted by the condition that maximum internal cycle length t_p^{max} must be significantly larger than the time for a ping wave to propagate from one end of the swarm to the other. It ensures that ping waves likely propagate through the entire system without colliding with other waves, thus enabling swarm-wide communication. For example, as shown in Figure 3.20, in the primitive "estimating the number of swarm members", the performance of the estimate is affected by the choice of t_p^{max}. Now, considering a case where t_{ee} is high due to the employment of a slow communication mechanism for agent-to-agent communication, a high $t_e e$ will necessitate a high t_p^{max}. In such a case, there needs to be a trade off between the number of agents and operational time scales. However, more sophisticated techniques can be employed in addition to the simple ping counting mechanism used in

Section 3.3 for estimating the number of swarm members. Brambilla et al. (2009) suggests a variation of Melhuish et al. (1999) to more efficiently estimate the number of members in a swarm by changing the cycle length during the run time. Such mechanisms can be used to increase the efficiency of the primitives and improve their scalability.

3.7.2 Design considerations

Directives for parameter selection for primitives in general have already been discussed in Section 3.5 and in Section 3.2. The selection of parameters is crucial for successfully using the WOSP paradigm. However, there is a wide range of parameters for which the primitives function and a set of parameters can be reliably found for each primitive. The optimality of the selected parameter set needs to be evaluated based on the operating time scales and available hardware by the swarm programmer. For example, if the cycle length of the swarm is large enough, there will be less collisions of pings and therefore, there can be a more reliable count of the number of members in the swarm. If the time scales of operation demand a faster convergence for a given number of agents, then the programmer must resort to an alternate method for counting the number of members in the swarm. However, an analysis paying attention to physical parameters for a prospective implementation of WOSP in a robotic swarm is beyond the scope of this thesis.

One of the prerequisites for a swarm to be able to implement WOSP is directional communication similar to most animals in nature which exhibit swarm behavior. In this chapter, most simulations conducted follow the assumption that agents have the ability to precisely detect the direction of incoming pings. In practice, this requirement can be substantially loosened for the agents to have a lower angular resolution without significant loss of functionality as shown in Figure 3.22. In other words, a rough perception of the direction of the incoming ping is sufficient for the basic functionality of WOSP primitives that depend upon incoming ping direction. More specifically, primitives involving motion or directionality such as aggregation, gas expansion, collective motion, localizing an object will suffer in preciseness with a lower resolution of angular perception. For primitives that requires the agent to move in a particular direction, low resolution can be compensated with slower movement speeds.

The development of WOSP is closely associated with the project subCULTron (subCULTron, 2015), a project aiming at deploying a heterogeneous swarm of underwater robots to monitor environmental parameters in the lagoon of Venice. Within the framework of this project, individual primitives of WOSP are already being used for swarm control. Robotic systems such as subCULTron, which employ a large number of individual agents in a noisy environment aiming for autonomous operation, can benefit from WOSP. A practical application of the subCULTron swarm is shown in Chapter 6. In a real world implementation of WOSP, simplifying assumptions employed in this chapter will need to be addressed. For example, the speed of the communication mechanism will determine the speed with which a message travels through the swarm. The physical speed of the agents will need tuning with respect to communication speed in order to keep the swarm together. A number of other practical considerations will need to be addressed depending on the specific physical properties of the swarm. In the experiment conducted in Section 3.6, this concern was not a major concern since it was not physically possible for thymio-II robots to be faster than the communication speed of the blue-light modules. In general, WOSP unifies several useful swarm behaviors under one paradigm. However, the simplifications made in this chapter to demonstrate the capabilities of WOSP need to be addressed by the programmer for each individual swarm.

Conclusively, a general unifying paradigm for common swarm behaviors which can be used to control a swarm of agents is presented in this chapter. As demonstrated through the individual primitives in this chapter, the 1-bit bioinspired communication paradigm can be used in swarm robotics to produce a large variety of behaviors for simple robotic systems with limited capabilities and minimalistic communication. The basic primitives can also be combined to form complex behaviors using the same underlying minimalisitic paradigm.

4 Collective Emergent taxis

In the previous chapter, I discussed various collective behaviors a swarm can inherit by following a simple, minimalistic and bio-inspired communication paradigm. In Section 3.4, examples are demonstrated as to how simple primitives can be combined to accomplish complex collective tasks. In this chapter, I showcase an alternate option for using emergent tendencies to accomplish swarm wide gradient "taxis" (Webb, 1998; Grodzicki and Caputa, 2005) behavior. This chapter is based on the following publication:

- Varughese, J. C., Thenius, R., Wotawa, F., and Schmickl, T. (2016). FSTaxis algorithm: Bio-inspired emergent gradient taxis. In *Proceedings of the 15th International Conference on the Synthesis and Simulation of Living Systems*. MIT Press

"Taxis" is a term borrowed from the field of biology where it is defined as the motion of an organism in response to a particular stimuli (Kendeigh, 1961). In the context of swarm robotics and for this thesis, taxis refers to the motion of an agent or a robot in response to a measured quantity (Kengyel et al., 2011; Varughese et al., 2016). In contrast to combining primitives in a sequential manner to accomplish taxis (as shown in Section 3.4), aggregation behavior can be performed while modulating the cycle time (based on some measured physical quantity) of the agents in parallel. When repeated, the collective moves towards the global maximum of the physical quantity being measured and used for modulating the maximum cycle time. This collective resultant movement to the global maximum happens provided there is a gradient leading to the global maximum. We call this algorithm based on WOSP behavior the "Firefly and Slime mold Taxis" or the "FSTaxis" algorithm.

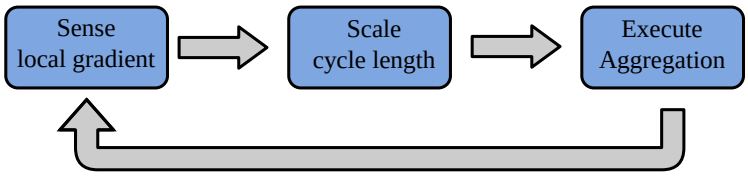


Figure 4.1: Figure shows the overall scheme of the FSTaxis algorithm where the individual agents measure the local value of the physical quantity being measured, scale the cycle length accordingly and then execute the aggregation primitive by moving towards incoming pings.

4.1 FSTaxis: a WOSP based taxis algorithm

The "FSTaxis" (Firefly and Slime mold Taxis) algorithm is an emergent gradient taxis algorithm based on the paradigm introduced in Section 3.2. In addition to the agent capabilities mentioned in Section 3.2, the agent has an on-board sensor which can measure a physical quantity. The aggregation primitive is repeatedly executed and the value of the individual agent's maximum internal timer value t_p^{max} is reset according to the measured sensor value to ascend a gradient. A schematic diagram of agent behavior is shown in Figure 4.1. A more specific explanation of the FSTaxis algorithm is given below and is categorized into communication and motion behaviors of the individual agents.

The communication behavior of the FSTaxis algorithm is the same as that of WOSP as introduced in Section 3.2 and naturally, the parameters used in Section 3.2 will be used here as well. In order to build an emergent gradient taxis, each agent sets its cycle length (t_p^{max}) modulated by the measured value of the environmental gradient at its position. If the internal timer of any agent counts down to zero before a ping is received, the agent broadcasts a ping and resets its own cycle length t_p^{max} by associating it with the measured value, g_p, at its position. Following the communication behavior of WOSP, that "original" ping is further relayed by the neighboring agents. In order to provide scaling of pinging frequencies to meaningful values, two preset maximum and minimum are selected for the gradient under consideration: g_{max} and g_{min} respectively. Equation 4.1 shows the relation between the ping frequency of agents and the inherent cycle time of agents. α and ω are constants with unit of time (s) and are selected in a manner that α corresponds to the minimum t_p^{max} possible and ω provides resolution for

meaningful corresponding change in t_p^{max} according to changes in the values of the measured environmental quantity. The values of constants used are the same throughout this chapter and is shown in Table 4.1. Here, the physical quantity being measured is considered to be of unit u.

$$t_p^{max} = \frac{1}{f_p} \tag{4.1}$$

$$f_p = \alpha + \frac{((g_p - g_{min})}{(g_{max} - g_{min})}\omega \tag{4.2}$$

As for movement, an agent being in an inactive mode does not move. As shown in Figure 4.2, motion is initiated in the active mode. When any agent receives a ping it sets itself to active mode, sets its own heading towards the received ping and moves a fixed step of length, d. A ping can only be perceived within the limited sensor range, r, of the agent, therefore limiting the number of agents that are able to influence any particular agent. In the scenario described above, it is possible that each agent receives multiple pings from different directions. In such as case, the agent will calculate the mean heading and will move a distance d in that particular direction. If an agent's internal clock triggers, it transitions into the active state, sends out a ping but does not move in that particular cycle.

When a swarm of agents executes the FSTaxis algorithm as per description above, scroll waves of pings similar to that in slime mold propagate through the swarm as in the case of all primitives presented in Section 3.3. In contrast to the primitives in Section 3.3, the t_p^{max} is not same for all agents. Since the internal timer of the agent with the highest gradient value will count to zero first, it pings first. Therefore, the direction of the wave will be from areas with higher values towards areas with lower gradient values. When an agent receives a ping or multiple pings, it will move towards the mean direction of the incoming pings. Since the agents whose internal timer triggers ("initiator") do not move, the other agents will gather around the initiator. When the agents are in their new position, their internal clock values are reset to values of the environmental value. Whichever agent's internal clock triggers first becomes the new "initiator" and the swarm then gathers around this agent. This repeated execution of the aggregation primitive results in an emergent gradient taxis. The pseudo code of the algorithm can be found in Algorithm 4.1.

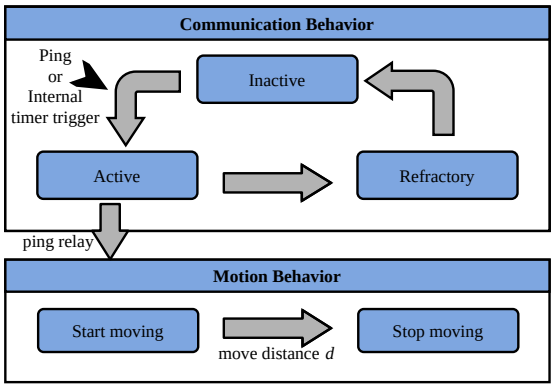


Figure 4.2: The state transition diagram of the FSTaxis algorithm is shown here. The agents can be in three states: active, refractory and inactive. An agent transitions into the active state when it receives a ping from a neighboring agent or when its own internal clock triggers. After broadcasting the ping, the agent transitions into a refractory mode. After the refractory time, the agent transitions into the inactive mode. The active mode triggers motion behavior and the agent takes a preset step of length d in the average of the direction of all incoming pings. After the motion is complete, their internal clock values are reset to values of the environmental value according to Equation 4.1.

Constants							
	ω	t_{ref}	r	d	α	g_{max}	g_{min}
Value	0.1	5	3	0.5	0.008	50	5
Units	s	s	p[1]	r	s	u	u

Table 4.1: Table showing all parameters used in the FSTaxis algorithm.

4.2 Gradient ascent with FSTaxis

To demonstrate the gradient ascent capability in simulation, a linear gradient represented by Equation 4.3 and a hyper ellipsoid gradient represented in Equation 4.4 are used. Noisy variants of these gradients are also used as

[1] unit p in Table 4.1 represents distance unit in Netlogo

```
Data: Paradigm parameters
Result: -
state ← inactive;
timer(t_p) ← random integer ∈ (0, t_p^max];
while primitive do
    decrement timer(t_p);
    if agent in refractory state then
        wait for refractory_time;
        if refractory_time is over then
            state ← inactive
    if agent in active state then
        broadcast ping;
        state ← refractory
    if agent in inactive state then
        listen for incoming pings;
        if ping received then
            state ← active;
            execute Relay-CodeBlock;
    if timer(t_p) ≤ o then
        state ← active;
        execute Initiate-CodeBlock;
Function Initiate-CodeBlock
    timer(t_p) ← t_p^max ;
Function Relay-CodeBlock
    timer(t_p) ← t_p^max;
    record ping direction;
    Calculate average of incoming pings;
    move d steps towards incoming ping;
    f_p ← f(local gradient value) ;
```

Algorithm 4.1: Pseudo code for the FSTaxis algorithm

test functions in order to test the FSTaxis algorithm. All the constants used in this experiment are shown in Table 4.1.

4.2.1 Simulations without noise

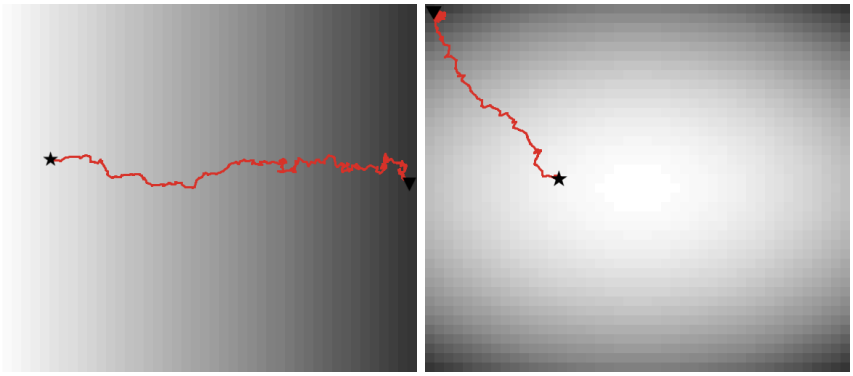
Figures 4.3(a) and 4.3(b) show the simulation environment setup with both a linear and a hyper ellipsoid gradient. The color scaling represents the gradient value of the environment. The goal of the gradient ascent algorithm will be to go towards the dark colored areas. The red line represents the trajectory of the mean position of the swarm from the starting point (represented by the star) to convergence (represented by the inverted triangle). The swarm is considered to have converged when its mean position oscillates around the area with the maximum gradient value. The trajectory shown is the result of one of the exemplary runs from the 100 runs conducted with the test gradients. 100% of the runs resulted in convergence to the maximum gradient value.

$$f(x) = x \tag{4.3}$$

The axis parallel hyper ellipsoid, represented by its standard equation 4.4, is a convex, continuous function and multiple modal function.

$$f(x) = \sum_{i=1}^{2} x_i^2, where - 5.12 < x_i < 5.12 \tag{4.4}$$

For the hyper ellipsoid gradient, there are four goals at the corners of the arena with the highest gradient value. The area of the goal (corners of the ellipsoid) is merely 0.23% of the total area of the arena. Therefore, random chances of the swarm converging to the goal are minimal. Figure 4.3(b) shows the FSTaxis algorithm tested with a hyper ellipsoid gradient, the thick red line shows the trajectory of the swarm. The black star marks the starting point and the inverted triangle shows the area of convergence. The trajectory shown is the result of one of the exemplary runs from the 100 runs conducted with this test gradient. 100% of the runs resulted in convergence to the maximum gradient value.



(a) Linear Gradient (b) Hyper ellipsoid gradient

Figure 4.3: Behavior of the FSTaxis algorithm to two gradients. (a): A linear gradient was presented to the FSTaxis algorithm. (b): An axis parallel hyper ellipsoid gradient was presented to a swarm executing FSTaxis algorithm. In both (a) an (b), the gray-scale shows the gradient value and the goal is the darkest area; the star symbol represents the starting point and the inverted triangle marks the point where the swarm converged.

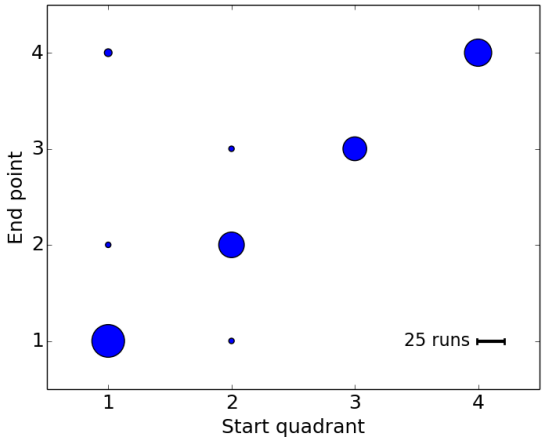
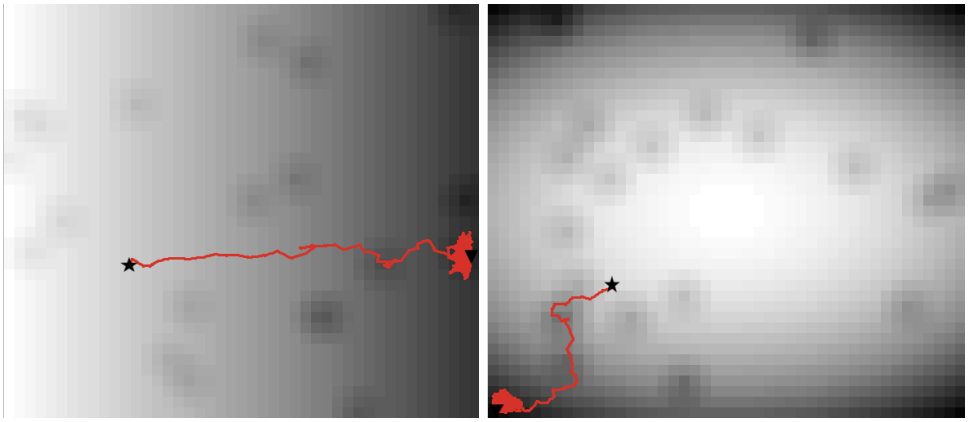


Figure 4.4: Bubble plot showing the region of convergence relative to the starting point of the swarm for 100 iterations. The reference line in the right lower corner shows the diameter of a 25 run bubble. The X-axis shows the quadrant in the arena where the swarm started and the Y-axis shows the region of convergence. Numbers 1, 2, 3, 4 refer to the quadrants as referred to in the Cartesian coordinate system. It is seen that in 95% of the runs, the swarm converges to the goal nearest to it.

In case of Figure 4.3(b), it can be seen that the swarm starts at the center and has the possibility of ascending four different gradients towards four corners. Figure 4.4 shows the region of convergence relative to the starting point of the swarm for 100 iterations. The X-axis shows the quadrant in the arena where the swarm started and the Y-axis shows the region of convergence. It is seen that in 95% of the runs, the swarm converges to the goal nearest to it. The 5 % errors are attributed to the fact that, when the swarm starts at the midpoint between two gradients, it has to choose which gradient to ascend. This decision depends on which agent's internal clock triggers first and hence, is dependent on agent placement and initial conditions.

4.2.2 Gradients with local optima

In order to test the ability of the FSTaxis algorithm to overcome small local optima, 20 randomly generated obstructions or "hills" have been introduced to the smooth gradient. These obstacles attract them to stay at these local optima if sufficient exploration is not introduced. Figure 4.5(b) and 4.5(a) show the result of a random successful attempt out of the 10,000 iterations of FSTaxis algorithm run with noisy hyper ellipsoid gradient and linear gradient respectively. As shown in Figure 4.6, the simulations with noisy gradients were conducted with varying steepness of local optima and spread of each optima. The number of obstructions were kept constant at 10. For each obstruction spread ranging from 1 to 10 and steepness ranging from 1 to 3 times the normal gradient, 100 iterations were run to observe the convergence to the goal.



(a) Linear Gradient with noise (b) Hyper ellipsoid gradient with noise

Figure 4.5: Behavior of the FSTaxis algorithm to two noisy gradients. (a): An axis parallel hyper ellipsoid gradient with numerous local maxima was presented to the FSTaxis algorithm. The gray-scale shows the gradient value and the goal is the darkest area; the star symbol represents the starting point and the inverted triangle marks the point where the swarm converged. (b): An axis parallel hyper ellipsoid gradient with numerous local maxima was presented to the FSTaxis algorithm. The gray-scale shows the gradient value and the goal is the darkest area; the star symbol represents the starting point and the inverted triangle marks the point where the swarm converged.

From figures 4.5(a) and 4.5(b), when the agents executing FSTaxis algorithm are presented with obstructions in the gradient, they are able to overcome local optima introduced. As individual agents move towards the leader, they overshoot the leader (agent whose internal clock triggered a ping) and escape the local optima. Figure 4.6 shows a graph representing the change in rate of convergence rate with the steepness of the local optima introduced. It is seen that for 20 obstacles in the arena, when the size of local optima (R_{optima}) is below one sensor range r, that is $R_{optima} < 1r$, 100% of the runs converge to the goal. As empirically shown in Figure 4.6, as the size and spread of local optima rises, the rate of convergence decreases.

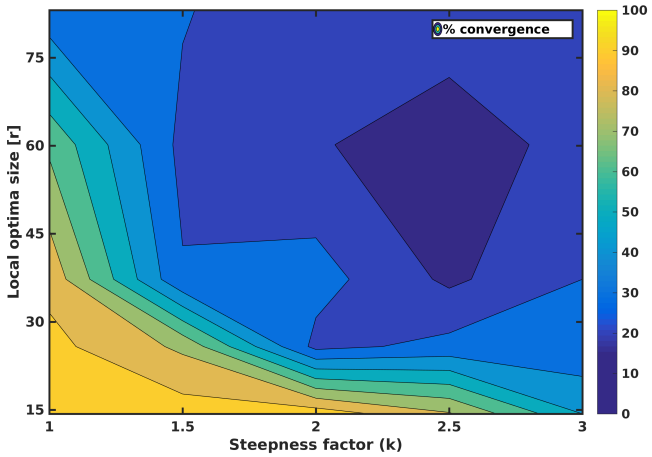


Figure 4.6: The color map shows the percentage convergence in presence of noise. A set of 10 obstructions or local optima were introduced into each gradient. The X-axis represents the factor of multiplication of steepness of obstructions with respect to the normal gradient. The Y-axis represents the area covered by each of these 10 obstructions. For each set of hill size and steepness, 100 iterations were run. Yellow colored areas show 100% convergence. It is seen that as for a fixed sensor range r and distance traveled with each ping d, the percentage of convergence largely depends upon how steep and how large the local optima is.

4.3 Summing up the FSTaxis algorithm

By calling the taxis algorithm emergent, I point out that the algorithm does not explicitly seek to do taxis but the taxis behavior is a property that is more than the sum of its parts. While emergence does not have an all encompassing accepted definition, the commonly accepted definition is *"the whole is more than the sum of parts"*. Alternatively in Bonabeau and Theraulaz (1994), the authors define emergence as *"a process through which entirely new behaviors appear, whose properties cannot be derived from a given model of how the system behaves, so that another model has to be built in order to deal with these new behaviors"*. Since the agents do not actively compare their current gradient value and the previous gradient value but merely move towards the incoming ping, the gradient ascent of the agents is an emergent phenomenon.

In addition to the gradient ascent being an emergent property, since there is no comparison of gradient values, it can be said that the FSTaxis algorithm remains strictly temporally and spatially local. In order for FSTaxis to work, agents have to be merely informed about the presence of other agents in their sensor range. Therefore, it is only required of the agent to sense the gradient value at current position, adjust the agent's own ping behavior accordingly and broadcast a single-bit ping to make its presence known. Since the algorithm uses a single-bit ping as in the case of WOSP, it is minimalistic in its communication requirements. It also inherits all the properties of WOSP described in Section 3.7.

In Section 4.2, it has been shown that the FSTaxis algorithm is able to work with multiple local optima. As seen in Figure 4.6, in presence of local optima that are steep enough, the FSTaxis algorithm is likely to get stuck in the local maxima. Drawing from these results, it can be concluded that while FSTaxis is able to ascend simple gradients with noise, they do not guarantee the convergence to the global maxima in case of multi-modal gradients. One way to combat the swarm getting stuck in a local maxima is to increase the spread of the swarm so that some agents in the swarm reach the area where local conditions are more optimal than those in the local optimum. In such a case, these agents will be able to pull the agents out of the local optimum. Figure 4.7 shows the different spreads of swarms (R_{rms}) over run times for various step sizes d for the agents while keeping the sensor range r constant. Here, R_{rms} is the root mean square distance of each agent from

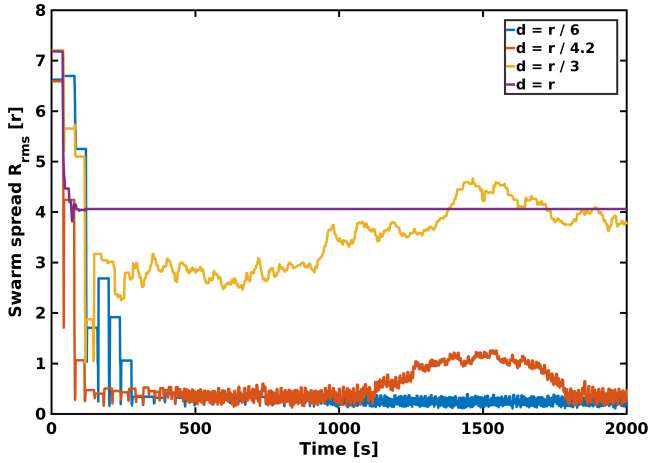


Figure 4.7: The change of the spread of the swarm represented by R_{rms} during four different runs. It can be seen that when d, the step size of each agent is low, the swarm stays coherent. As d increases, the swarm spreads out too much due to the large step sizes taken. Therefore, d has to be so selected with continuous connectivity of the swarm in mind.

the center of mass of the swarm. It can be seen that for small step sizes such as $d = \frac{r}{6}$ and $d = \frac{r}{4.2}$, the swarm remains coherent, that is, the agents maintain their connectivity with one another. This is shown by the swarm spread R_{rms} remaining within a single sensor range r. However, for larger step sizes $d \geq r$, although the spread of the swarm increases as shown by the rise in R_{rms}, the swarm loses its coherence, that is, the agents lose connectivity with each other due to large distances traveled based on single pings. Therefore, the spread of the swarm can be increased partially by increasing the step sizes but beyond a limit, the swarm loses its coherence. Other ways of increasing the spread of the swarm are to add more agents and to use communication devices that have an increased range, r.

Through this chapter, it is demonstrated that FSTaxis algorithm is a an emergent solution for gradient ascent using WOSP in a swarm of agents. The minimalism FSTaxis inherits from the WOSP communication paradigm makes it attractive because it requires only a single-bit communication between the agents. Additionally, the FSTaxis algorithm works based on simple rules and requires basic hardware.

5 A Resilience Case Study

One of the most fascinating aspects of swarming behavior is their high tolerance towards the loss of individual entities without losing the overall performance of the superorganism. 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 (Angerer et al., 2015). In this chapter, I will explore the resilience of the WOSP paradigm to communication failures. As detailed in Appendix A, this chapter is based on the following publications:

- Varughese, J. C., Moser, D., Thenius, R., Wotawa, F., and Schmickl, T. (2019b). *swarmFSTaxis: Borrowing a Swarm Communication Mechanism from Fireflies and Slime Mold*, pages 213–222. Springer International Publishing, Cham
- Varughese, J. C., Thenius, R., Schmickl, T., and Wotawa, F. (2017). Quantification and analysis of the resilience of two swarm intelligent algorithms. In *GCAI 2017. 3rd Global Conference on Artificial Intelligence*, volume 50 of *EPiC Series in Computing*, pages 148–161. EasyChair

5.1 Resilience of swarms

Several studies have been conducted about the aforementioned robustness of swarm intelligent algorithms. In Kengyel et al. (2016), the authors presented an analysis of the BEECLUST algorithm and experimentally verified the robustness of the algorithm by adding agents with impaired temperature sensors. In Bjerknes and Winfield (2013), the authors performed a comprehensive analysis of the resilience of the swarmtaxis algorithm with respect to failure modes such as IR sensor failure and motor failure. In this chapter, I will explore the resilience of the WOSP behavior using "taxis" behavior introduced in Chapter 4 as a case study. I consider "resilience" as

the ability of a system to perform its overall goals even with sub-optimal agent behavior. In order to approach the topic of resilience of the algorithms in a comprehensive manner, all the capabilities of the individuals of a swarm and how their failure could affect the overall goals of the swarm need to be considered. 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 the performance of the swarm in different ways. Since this thesis presents a paradigm for communication, I will concentrate on agent-to-agent communication failures and their effects on the overall swarm behavior in this chapter.

This resilience study is part of the project subCULTron (subCULTron, 2015) which aims at developing a swarm of autonomous underwater robots to perform environmental measurements and monitoring. One of the sub-tasks 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 this chapter deals with the effects of communication behavior on the overall swarm performance, I have to clarify what kind of communication is available on the robotic platforms being considered. 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 several robots need to communicate with one another. The robots in subCULTron are therefore equipped with local communication modes such as blue-light communication, where small packets of modulated blue-light signals are exchanged. 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 in mind, our algorithms and tasks must be resilient to failure in agent-to-agent communication. The rest of the chapter is dedicated to selecting two swarm intelligence algorithms and subjecting them to a resilience test.

5.2 Relevant algorithms

Algorithms for collective navigation and foraging problems are well investigated fields in swarm intelligence (Tan and Zheng, 2013; Ducatelle et al.,

2014; Senanayake et al., 2016). Many of these algorithms that are inspired by natural swarms can be broadly subdivided into pheromone based navigation (Sugawara et al., 2004), navigation based on physical robot chains (Maes et al., 1996), navigation based on signaling (Bjerknes et al., 2007; Varughese et al., 2016) and navigation based on pheromone-like gradients (Schmickl and Crailsheim, 2008; Hoff et al., 2010). In Sugawara et al. (2004), the authors used a combination of cameras and LEDs to project virtual pheromone trails which "evaporate" over time. Such global observation based algorithms are evidently not suitable for an underwater environment where global observation is expensive and difficult to implement. In Maes et al. (1996), the authors used 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 unfeasible for subCULTron and similar underwater projects. In the approach used by Schmickl and Crailsheim (2008) and Hoff et al. (2010), the authors presented 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 Bjerknes et al. (2007) and Varughese et al. (2016), the authors used 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, approaches which employ a single-ping communication between agents are examined.

The underwater environment introduces additional constraints over and above the widely accepted swarm intelligence criteria (Turgut et al., 2007). 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 (Bjerknes et al., 2007) and the FSTaxis (Varughese et al., 2016) algorithm fulfil 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 every agent in the swarm while in FSTaxis, the agents need to be indirectly connected to the rest of the swarm through at least one neighbor. Despite few differences, largely, the algorithms are comparable to each other, thus enabling us to use the performance parameters I designed for this comparison. Since the FSTaxis algorithm has already been introduced in Chapter 4, I will only detail the swarmtaxis algorithm in this section.

5.2.1 The swarmtaxis algorithm

The swarmtaxis algorithm (Bjerknes et al., 2007) 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 smaller than those in illuminated mode. This means that an agent in illuminated mode can detect agents in shadowed mode 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.

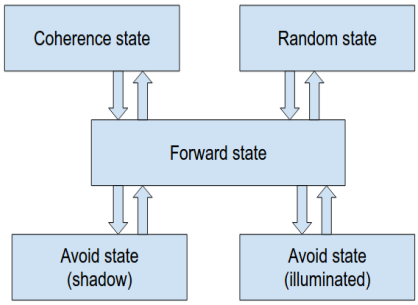
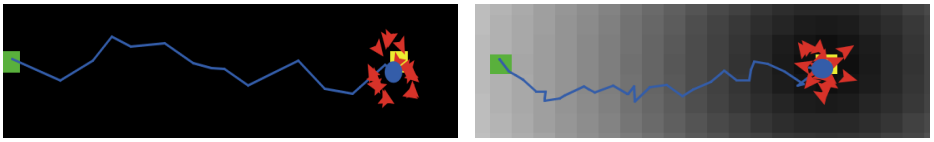


Figure 5.1: State transition diagram of the swarmtaxis algorithm (Bjerknes et al., 2007).

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 to the agent it detected within its avoid radius and then enters the "forward" state.

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 sensed from afar 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 the shadowed agents, which is in fact, the direction of the goal. A typical run in the swarmtaxis algorithm is shown in Figure 5.2(a). As discussed above, the "coherence" state of the swarmtaxis algorithm keeps the agents together. After the initial publication (Bjerknes et al., 2007), the authors presented a method where the entire swarm needed to be connected for the swarm to consistently move towards the goal without entering the "coherence" state. In later modifications of the swarmtaxis algorithm (Winfield and Nembrini, 2012; Bjerknes and Winfield, 2013), the authors presented an improved version (β and ω versions) of the algorithm, where the agents were required to communicate more than a single ping in order to work reliably. In this chapter, I will therefore consider the basic algorithm



(a) (b)

Figure 5.2: Typical runs of the swarmtaxis and the FSTaxis algorithms are shown in figures 5.2(a) and 5.2(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 5.2(b) represent the local gradient value.

presented in Bjerknes et al. (2007), and based on the connectivity study presented in Winfield and Nembrini (2012), I will present the improvement in resilience due to the relaxation of the connectivity constraint α.

5.3 FSTaxis vs swarmtaxis: Comparing resilience

In this section, the simulation of failures in the algorithms of interest and the performance measures used to study the effects of agent-to-agent communication failure in each of these algorithms are described. The parameters used for simulation are the same as those used in Chapter 4 and Bjerknes et al. (2007) for FSTaxis and swarmtaxis respectively. In order to ensure fair comparison, a common communication range and agent velocity has been used for agents in both algorithms. Initially, the agents are distributed uniformly around the starting point. 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.

5.3.1 Simulating failures

In Section 5.2, I discussed each algorithm and saw that in both of these algorithms, agents use a single-bit communication method to let the sur-

rounding 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 5.3 and 5.4 respectively. In order to simulate this failure, I used a probability based roll of a dice each time the agent attempted to communicate and decided whether the communication should fail or not. Then, this failure probability was increased progressively and, for each failure probability, I collected 100 data sets in order to gain substantial data to support our conclusions. In the following section, I will discuss data collection during each simulation run and why each parameter is suitable to analyze the performance of the swarm.

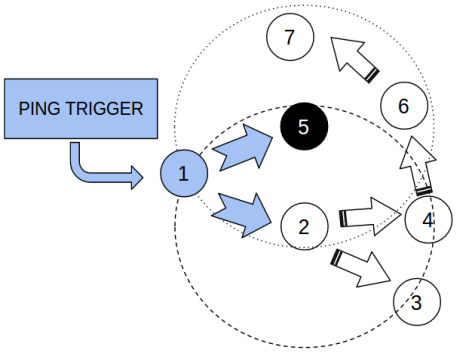


Figure 5.3: A scenario showing a ping failure in the FSTaxis algorithm. Agent 1, triggered by its internal counter, broadcasts a ping. The arrows represent the relaying of that ping to those agents whose communication device can detect this ping (represented by dotted circles for agents 5 and 2). Agent 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.

Ping failure in FSTaxis

Figure 5.3 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

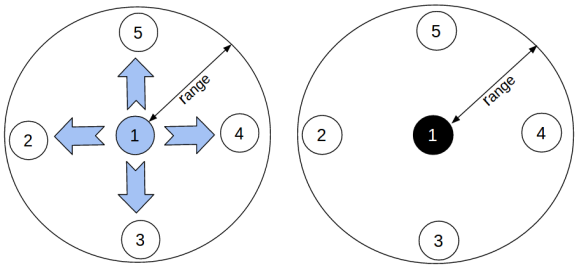


Figure 5.4: Two scenarios showing a successful and a failed communication in the swarmtaxis algorithm. Scenario 1 is shown on the left, where agent 1 broadcasts its periodic ping and all other agents in the range perceive this ping. Scenario 2 is shown on the right, where agent 1 (black) has a ping malfunction which prevents it from broadcasting the ping, and hence, the other agents in range are blind to the presence of agent 1.

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.

Ping failure in swarmtaxis

Figure 5.4 illustrates two scenarios of ping success and failure in the swarmtaxis algorithm. In Figure 5.4(a), a successful scenario of pinging agents within a range is shown. The circle represents a range within which all agents can communicate with each other. In Figure 5.4(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 the number of connected agents.

5.4 Performance measures

In this section, I describe two 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.

5.4.1 Time performance

One of the intuitive performance measures is the time or the number of simulation ticks (defined in unit s in Section 3.2) the swarm takes to "converge" to the goal. Here, I define convergence as the centroid (or centre of mass) 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 2000 s for the FSTaxis algorithm and 10000 s for the swarmtaxis algorithm. The standard deviation of the number of ticks for each of these algorithms does not exceed 200 s. The difference in the average of ticks does not imply worse performance as the simulation time is also 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 simulation ticks typically needed for convergence, I limited the simulation time to 10 times the average ticks a swarm needs to converge to the goal with all functions intact. Thus, the simulation time is taken to be 100000 s for the swarmtaxis algorithm and 20000 s for the FSTaxis algorithm. Therefore, the time performance, normalized against its own ideal performance, can thus be represented as per Equation 5.1.

$$time_{performance} = \frac{number\ of\ ticks}{maximum\ simulation\ time} \tag{5.1}$$

5.4.2 Optimal path and deviation

From figures 5.2(a) and 5.2(b), 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 connecting the starting point with 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 reveals some information about suboptimal swarm behavior. Following this logic, I consider the error between the actual path and the optimal path to be a performance measure of the swarm. Figure 5.5 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 freely drawn line traces the actual trajectory ρ of the centroid of the swarm and the ideal path can be represented as a vector $\overrightarrow{D}$. The trajectory ρ 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, I conducted 100 runs and obtained the set of all centroid trajectories, O_k, and, for each run, I obtained a set ρ_{kj} which contains points T_{kji}. Here, ρ_{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 T_{kji} on $\overrightarrow{D}$ was computed. Furthermore, the error vectors, obtained as shown in Equation 5.5, 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 5.6. 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 5.7, can be represented on a box-plot to visualize how the window of operation shifts with changing p_k.

$$\begin{aligned} P &= \{p_k \mid p_k = 0.05, 0.1, 0.15 \ldots 1\} \\ &\quad , \textit{ where } k = 1, 2, 3, \ldots, \mid P \mid \end{aligned} \tag{5.2}$$

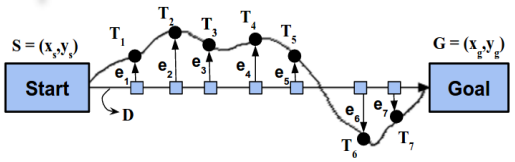


Figure 5.5: Illustration of the deviation of a swarm centroid trajectory from the straight line passing through (x_s, y_s) and (x_g, y_g). The "start" block represents the starting point of the swarm and the "goal" block represents the end point of the swarm. The freely drawn line represents the actual 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. shows the distance of points on T to the corresponding points on D.

$$\begin{aligned} \forall\ p_k \exists\ O_k,\ where\ O_k &= \{\rho_{kj} \mid j = 1, 2, \ldots, 100\} \\ and\ \rho_{kj} &= \{\overrightarrow{T_{kji}} \mid i = 1, 2, \ldots, N_{kj}\} \end{aligned} \tag{5.3}$$

$$\forall\ \overrightarrow{T_{kji}} \exists!\ \overrightarrow{D_{kji}} s.t.\quad (\overrightarrow{T_{kji}} - \overrightarrow{D_{kji}}) \perp (\overrightarrow{D_{kji}} - \overrightarrow{S}) \tag{5.4}$$

$$\overrightarrow{e_{kji}} = \overrightarrow{D_{kji}} - \overrightarrow{T_{kji}} \tag{5.5}$$

$$E_{kj}^{rms} = \frac{1}{N_{kj}} \sum_{i=1}^{N_{kj}} \|\overrightarrow{e_{kji}}\| \tag{5.6}$$

$$\forall\ p_k \exists\ \varepsilon_k,\ where\ \varepsilon_k = \{E_{kj}^{rms} \mid j \in [1, 2, \ldots, 100]\} \tag{5.7}$$

5.5 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 d = 0.5 units, where one unit is unit length in Cartesian coordinates.

5.5.1 Time performance

Figure 5.6 shows the performance parameter $t_{performance}$ and how it changes as the probability of failure increases. We see that $t_{performance}$ saturates as maximum simulation time is reached. Time performance saturates rapidly for the swarmtaxis algorithm, while the FSTaxis shows a wider range of operation before $t_{performance}$ saturates.

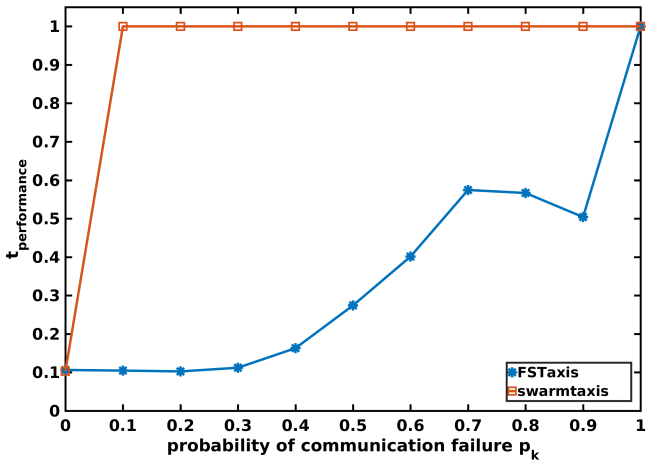


Figure 5.6: The performance parameter, $t_{performance}$ and how it changes as the probability of failure increases. The presented data is based on 1000 simulation runs (100 runs per p_k).

5.5.2 Root mean square error

Figures 5.8 and 5.9 show the distribution of mean square errors (E^{rms}) of the centroid trajectory with respect to the optimal path 'D' of a swarm executing both the FSTaxis algorithm and the swarmtaxis algorithm respectively. We see that as the probability of failure increases, the median and spread of E^{rms} increase 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

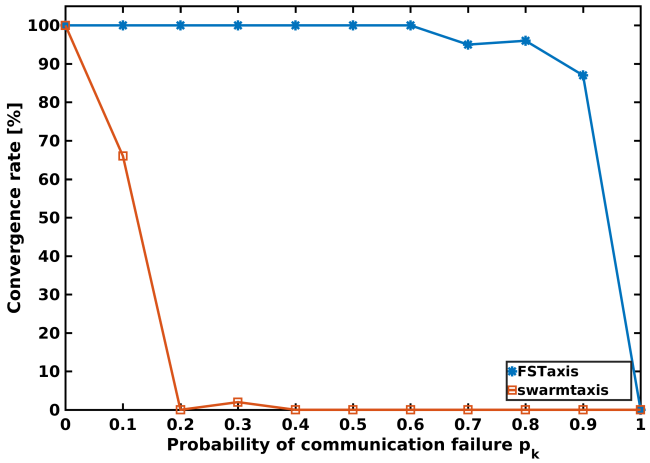


Figure 5.7: Percentage of runs of each algorithm that converged to the goal with increasing probability of failure. The presented data is based on 1000 simulation runs (100 runs per p_k).

"window of operation" (as defined in Section 5.7.2) widens as the ping loss increases for the FSTaxis algorithm. The probabilities with more than 50% non-converging runs are marked as "non-converging runs".

5.6 Discussion: swarmtaxis vs. FSTaxis

From figures 5.7 and 5.6, a "Resilient operating limit"(ROL) of failure probabilities p_k can be observed. In Figure 5.7, 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 a 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 Bjerknes et al. (2007), it can be seen 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

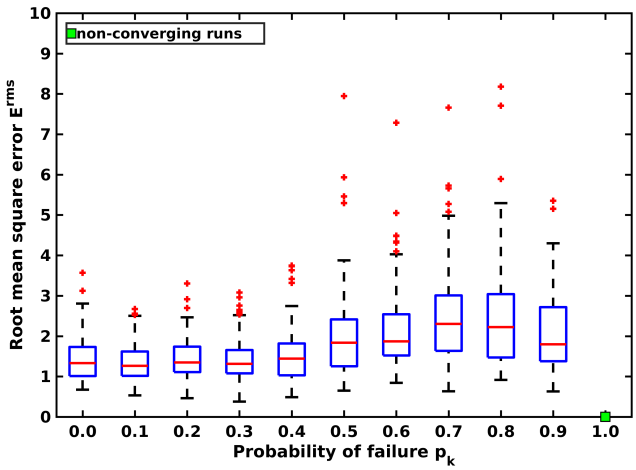


Figure 5.8: Distribution of root mean square error of centroid trajectory of a swarm executing FSTaxis. The probabilities of failure with more than 50% non-converging runs are marked as "non-converging runs". The red '+' signs represent the root mean square error of a single run among the total 1000 (100 runs per p_k) runs.

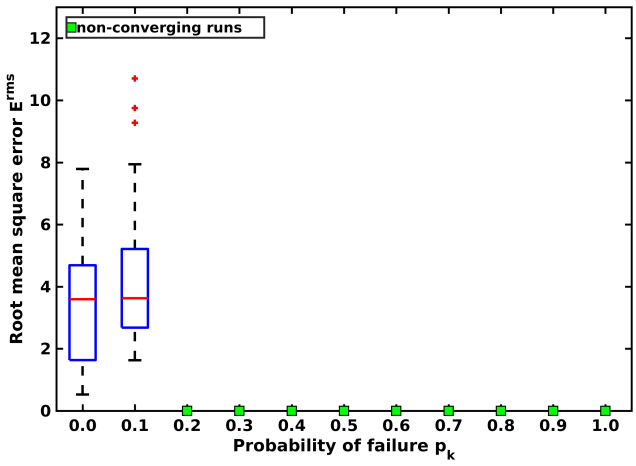


Figure 5.9: Distribution of root mean square error of centroid trajectory of a swarm executing swarmtaxis. The probabilities of failure with more than 50% non-converging runs are marked as "non-converging runs". The red '+' signs represent the root mean square error of a single run among the total 1000 (100 runs per p_k) runs.

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 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 number 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 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 Winfield and Nembrini (2012) and Bjerknes and Winfield (2013), the authors present a modified swarmtaxis algorithm and experiment with various α values, that is, they relax the connectivity criteria presented in Bjerknes et al. (2007). Such an approach will intuitively 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 which depends on the number of pings received.

From Figure 5.8, 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 5.2(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 5.3. As a result, the agents in the swarm either do not receive the pings or receive the pings via other agents whose communication mechanism works. 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 5.8) 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 5.7.

5.7 "swarmFSTaxis": making swarmtaxis more resilient

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 was to implement the communication strategy of the FSTaxis algorithm in swarmtaxis?". Section 5.7.1 introduces a modified swarmtaxis algorithm to employ the communication method of FSTaxis based on the research presented in the previous sections. In Section 5.2, it was established that the communication mechanisms of both the FSTaxis and the swarmtaxis algorithms have non-zero resilience to agent-to-agent communication failures while the FSTaxis algorithm exhibited a significantly higher resilience. Subsequently, I present how the resilient communication behavior of WOSP can improve the performance of the swarmtaxis algorithm. We refer to this hybrid between swarmtaxis and FSTaxis (with the communication behavior of WOSP) as the "swarmFSTaxis" algorithm.

5.7.1 The swarmFSTaxis Algorithm

In the swarmFSTaxis algorithm, additional to all capabilities described in Section 5.2.1, each agent is assumed to have internal timers and local directional communication. The goal can be occluded from an agent similar to the case of the swarmtaxis scenario. The behavior of the agents differ depending on whether they are "illuminated" or "shadowed". Figures 5.10 and 5.12 shows the state machine of the swarmFSTaxis algorithm.

As in the case of the FSTaxis algorithm described in Section 4.1, there are two types of behaviors in the swarmFSTaxis algorithm: the "ping" behavior and the "motion" behavior. The ping behavior describes the agent-to-agent communication during the execution of the algorithm. In ping behavior, as shown in Figure 5.10, the agents may assume three states: "pinging", "refractory" and "inactive". Initially, all agents are set to the inactive state. In the inactive state, the agent monitors its receivers for incoming single-bit local communication (pings). In the event of an incoming ping, the agent broadcasts a ping and enters the refractory state. During refractory time, t_{ref}, the agent is insensitive to all incoming pings. At the end of the

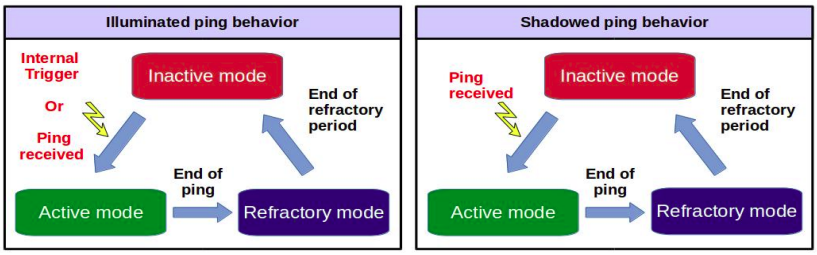


Figure 5.10: A state transition diagram of the ping behavior of the swarmFSTaxis algorithm is shown in the figure. Illuminated and shadowed agents differ in ping behavior in that the shadowed agents only relay pings while the illuminated agents both produce and relay pings.

refractory time, the agent transitions back to the inactive state. Apart from the above ping behavior, the "illuminated" agents have internal timers that are constantly counting down. When the timer counts down to zero, the agent broadcasts a ping. This means that an "illuminated" agent produces a ping either when the agent receives another ping or when its internal timer counts down to zero. The difference between the ping behaviors of shadowed and illuminated agents are illustrated in Figure 5.11.

In addition to the ping behavior described above, the agents also have a "motion" behavior. Unlike the ping behavior, the motion behavior is the same for all agents regardless of whether they are "illuminated" or "shadowed". There are two kinds of motion behavior: "general motion behavior" and "avoid motion behavior" as shown in Figure 5.12. In "General motion behavior", an agent at the event of an incoming ping, moves towards the incoming ping. In case there are multiple incoming pings, the agent moves towards the mean of the directions of all incoming pings.

During "avoid motion behavior", an agent moves away from a detected neighbor. As in the case of the parent algorithm, the swarmFSTaxis algorithm also implements dissimilar avoidance radii for "illuminated" and "shadowed" agents: $avoid_{illum}$ and $avoid_{shadow}$. The sensor range of the illuminated agents are set to a higher value which in effect, enables the agents to move away from an approaching shadowed agent.

Since the illuminated agents trigger pings when their internal timer counts down to zero, the waves will originate at the "illuminated" agents and propagate through the "shadowed" agents as they relay the pings. The

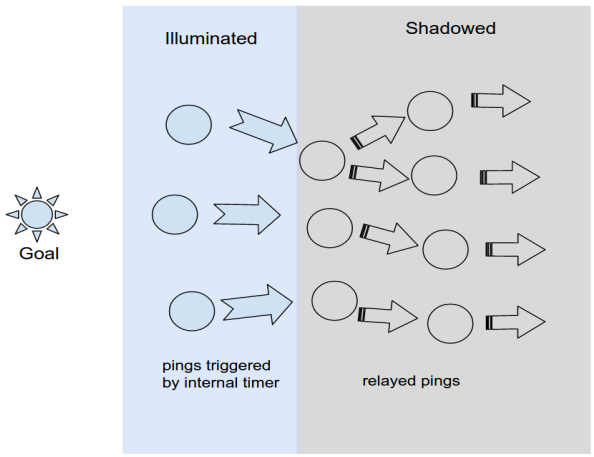


Figure 5.11: The ping mechanism of the swarmFSTaxis algorithm is illustrated in this figure. The "illuminated" agents have a low internal timer value and hence, will hijack the pinging frequency of the system by pinging frequently. The shadowed agents will keep relaying the pings produced by the illuminated agents.

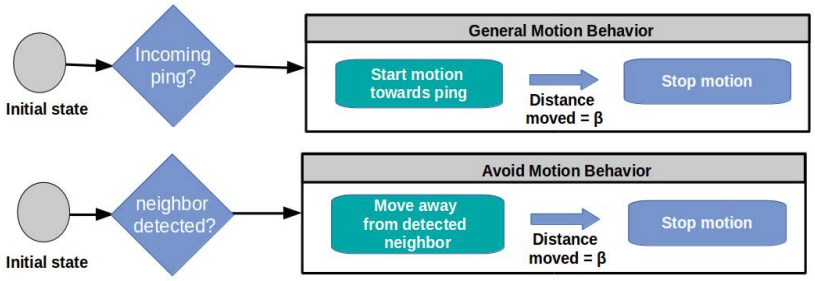


Figure 5.12: A state transition diagram of the motion behavior of the swarmFSTaxis algorithm is shown in this figure. Illuminated and shadowed agents have the same motion behavior triggered by either an incoming ping or a neighbor.

Table 5.1: Table showing all parameters used in the modified swarmtaxis algorithm.

Constants						
	t_{ref}	r	d	$swarm\_size$	$avoid_{illum}$	$avoid_{shadow}$
Value	3	2.5	0.1	21	0.28	0.16
Units	timestep	p[1]	r	-	r	r

"general motion behavior" ensures that the swarm stays together with the agents moving towards the incoming ping while the "avoid motion behavior" ensures that the illuminated agents move away from the shadowed agents and, in effect, move towards the goal.

5.7.2 Testing swarmFSTaxis

As in the case of Chapters 3 and 4, r is used as a unit representing one sensor radius of the agent. At the beginning of a typical run, the agents are distributed uniformly around a starting point and then, the algorithm is executed. The centroid of the swarm is used as a collective position estimate of the swarm. Once the centroid of the swarm reaches the goal as shown in Figure 5.13(b), the run is terminated. During the entire run, the position of the centroid of the swarm is tracked in order to produce a representative trajectory for the motion of the swarm as a whole in each run. The constants used for simulation are shown in Table 5.1.

In order to compare the modified algorithm with its parent, I will use the time performance measure introduced in Section 5.4. A set of 100 simulation runs has been conducted for each algorithm. To make the runs comparable, all runs were started from the same point, had the same swarm size and the same parameters as shown in Table 5.1 and had the same goal. Also, the parameters used such as sensor range of illuminated and shadowed agents, range of directional communication and distance moved during motion behavior were kept the same for all 100 runs. The time performance of each of these algorithms has been recorded and plotted in Figure 5.14 for the swarmFSTaxis algorithm and the parent algorithm. Later, in Section 6.5, Figure 5.14 is discussed in detail.

[1] unit p in Table 5.1 represents distance unit in Netlogo.

(a) (b)

Figure 5.13: The starting condition of a typical simulation run of the swarmFSTaxis is shown in Figure 5.13(a). Figure 5.13(b) shows a converged run. The green patch (occluded by the white circle in Figure 5.13(a)) shows the starting point, the white circle shows the centroid of the swarm and the yellow patch shows the predefined goal. The white trace in Figure 5.13(b) shows the trajectory of the centroid of the swarm. The yellow agents are the illuminated agents and the blue ones are the shadowed agents.

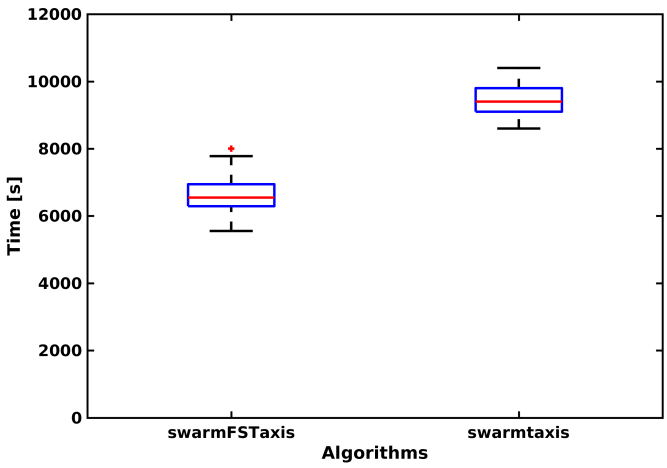


Figure 5.14: A box plot showing the simulation time each algorithm took to converge to the goal. The data from 100 runs of each algorithm are shown in the plot.

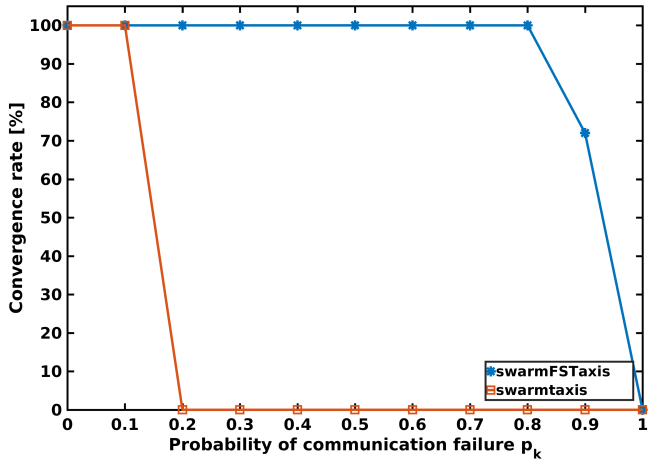


Figure 5.15: A plot showing the percentage of runs that converged to the goal for all failure probabilities.

For testing the algorithms for their resilience towards agent-to-agent communication failure, a probability of failure, $0 < p_k < 100$, was introduced to each ping that an agent broadcast. Before each agent broadcasts a ping, a roll of dice was performed with probability p_k to decide whether that ping should fail to be communicated. For each value of p_k, 100 runs were conducted and the simulation time taken by each algorithm to converge to the goal was recorded. Figure 5.15 shows the plot of the percentage of runs that converged to the goal for each failure probability. For both algorithms, 1000000 s were set as a limit beyond which a run was considered "non-converging".

5.7.3 Discussion: swarmFSTaxis

From Figure 5.15, I can conclude that the resilience of the swarmFSTaxis algorithm has increased dramatically as compared to the parent algorithm. This phenomenal increase in resilience of the algorithm is due to the fact that the algorithm does not depend on the count of pings for its state transitions but rather the direction of incoming pings. Even when as low as only a single ping reaches an agent, the agent performs general motion behavior. Meanwhile, in the case of the parent algorithm, if a ping from some of the agents fail, to be transmitted or received, the algorithm executes

sub-optimal transitions into the coherence state as it perceives that the swarm connectivity falls below the threshold α. The disadvantage of such an approach is that it makes the algorithm dependent on the success of communication of all the agents in the swarm. As shown in Equation 5.8, assuming the probability of failure of each ping as p_k, there is a collective probability, $p_{coherence}$, that an agent counting the pings makes a transition into the "coherence" state. In contrast, the swarmFSTaxis algorithm fails in its state transitions only when all the agents in the swarm or almost all agents in the swarm fail to transmit or receive a ping. This probability is of the order of $(\ p_k\ )^N$ where N is the number of members in the swarm and this is hence, very small for even very high values of p_k as seen in Figure 5.15.

$$p_{coherence} \ = \ p_k^1 \ + p_k^2 + p_k^3 \cdots + p_k^\alpha \tag{5.8}$$

In Figure 5.14 it can be seen that out of 100 runs of both algorithms, swarmFSTaxis algorithm is consistently faster than the parent algorithm. From Figure 5.14, the mean of the number of ticks to convergence for the swarmFSTaxis and the parent algorithms is 6597 and 9540 respectively. Therefore, the swarmFSTaxis algorithm has become about 30% faster than the parent algorithm. As explained above, fewer transitions into the "coherence" state helps the swarmFSTaxis algorithm to be faster than the parent algorithm. In a later publication (Winfield and Nembrini, 2012) the parent algorithm was improved and was made more resilient to such failures, however, the state transitions were still based on a poll based count. In contrast to this approach, the swarmFSTaxis algorithm uses the direction of the incoming ping instead of a poll based count to keep the swarm coherent.

From this chapter it is evident that due to the communication mechanism of WOSP, the FSTaxis algorithm exhibits resilient behavior even with a high ping loss. The reason for its high resilience is the ping relaying mechanism of WOSP as described in Section 3.2. 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 the “coherence” mode. In Winfield and Nembrini (2012), 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.

A modification of the swarmtaxis algorithm has been suggested and subsequent improvements in resilience have been demonstrated in this chapter. In the future, the WOSP communication mechanism has the potential to replace poll based counts that are common in engineered systems. Further research in this direction can ensure that the full capacity of such a communication mechanism is utilized.

6 WOSP for Event Detection

As the last chapter of this research, I present a practical application based on WOSP, which has so far been presented in simulation or in lab conditions. As detailed in Appendix A, this chapter is based on the following publication:

- Varughese, J. C., Hornischer, H., Thenius, R., Wotawa, F., and Schmickl, T. (2019a). Collective event detection using bio-inspired minimalistic communication in a swarm of underwater robots. Number 31, pages 634–641

6.1 Event detection

With the decreasing size of computation and memory devices, the number of computers has been increasing dramatically. Koh and Magee (2006) observed that computing power available per dollar has increased by a factor of ten roughly every four years over the last quarter of a century. The increase in the available computation power has brought massive parallel multi-agent systems to the forefront, such as ubiquitous computers (Kim and Follmer, 2017), IoT systems (Gubbi et al., 2013), swarm robotics (Zahadat and Schmickl, 2016; Witkowski and Ikegami, 2016). Such systems with increasingly large numbers of individual interacting parts pose challenges to the traditional top down control schemes. Therefore, decentralized computing strategies with little or no top down control are being widely explored and implemented (Schmickl et al., 2008; Cazangi et al., 2005). Inspiration for such strategies is drawn from self-organizing natural systems such as starling murmurations (Cavagna et al., 2010), honeybee colonies (Seeley, 1992) and slime mold aggregates (Durston, 1973; Bonner, 1949).

A decentralized algorithm is presented here for a swarm of underwater robots to detect, to collectively validate and to report significant variations in environmental parameters. In a nutshell, if a member of the swarm

detects an anomaly in its measurements it will register an event and alert its neighbors. A periodic oscillator and traveling wave based communication paradigm inspired by slime mold and fireflies is then used to periodically communicate with the neighbors who registered the event. As soon as the event is validated by a sufficient number of neighbors, an alert is sent to a base station.

The algorithm presented here is developed for a swarm of underwater robots which is intended to detect the *anoxia* phenomenon (Runca et al., 1996) in the lagoon of Venice. During anoxia the oxygen content of a small part of the lagoon decreases suddenly, resulting in the death of flora and fauna and thus, damaging the local ecosystem. Thenius et al. (2016) suggested a strategy for examining and documenting this recurring phenomenon by utilizing a swarm of underwater robots. According to this strategy, a team of underwater robots, known as "aMussels", will be used for monitoring a set of environmental parameters, including oxygen concentration levels, to detect the phenomenon within the framework of project subCULTron (subCULTron, 2015). On the one hand, the detection of anoxia by individual robots needs to be validated with a number of neighbors before a global alarm can be raised for greater reliability. On the other hand, underwater communication is expensive, noisy and therefore, exchange of information between the robots needs to be minimized. For this reason I will focus on detecting and validating the event of anoxia with a swarm of robots while considering the modes of communication available on the aMussel robots.

Many environmental monitoring systems, which use sensor networks to collect data in a large area, have focused on reducing the energy consumption in order to increase network longevity (Zhou et al., 2015; Kaur and Sood, 2017). However, most of the existing protocols and algorithms for underwater sensor networks focus on making the algorithms usable for a wide range of communication devices, especially for deep sea environments. By contrast, I focused on designing an event detection algorithm suitable for the low cost, narrow bandwidth and low payload communication devices used in subCULTron. Specifically, aMussels are equipped with three kinds of underwater communication devices. Modulated blue light communication and electric sense for extremely short ranges ($\sim$ 1 meter) as well as a low-frequency acoustic nanomodem for a comparatively longer range underwater communication ($\sim$ 100 meter). In addition to that, aMussel robots are not mere sensor nodes but have the capability of diving up and down in a water column. This enables the aMussels to dive up to the water surface

and report the occurrence of an event using ultra long range communication devices rather than forwarding packets to the sink nodes like traditional sensor networks do. Keeping these constraints and special capabilities of the aMussel robots in mind, I suggest an algorithm for event detection in autonomous swarms of robots in the following .

6.2 Related work

Many algorithms and protocols have been proposed and implemented to improve deep sea monitoring using underwater wireless sensor networks (UWSNs)(Zhou et al., 2015; Debont et al., 2012). Although many techniques used in the classical wireless sensor networks can be used for their underwater counterparts, communication in an underwater environment is especially challenging. Therefore, I will give a brief overview of some event localization schemes suggested for UWSNs.

Since underwater sensor networks might be mobile due to underwater currents, the communication protocol presented by Zhou et al. (2015) includes a "heartbeat" which periodically communicates with neighboring nodes and constructs a routing tree. Additionally, the system aggregates data and processes it locally to detect an event before forwarding the event to the sink node.

Debont et al. (2012) suggested a solution for event detection using a cyclic graph model. The solution optimally placed "beacon" nodes which act as location-aware references for surrounding nodes. In Debont et al. (2012) the authors showed that the intelligent placement of beacons reduces the number of sends required by 80 % as compared to a naïve placement. In case of an event, a message is forwarded to the beacon node which in turn acts as a buffer to collect more event messages from other nodes. Then, a batch of messages containing alarms is forwarded to the sink node. While such an implementation is helpful for underwater sensor networks in general, it requires elaborate routing protocols and non-minimal message lengths.

Repeated and periodic exchange of information is employed in the above implementations of event detection. While a "heartbeat" signal (Zhou et al., 2015) is important for dynamic deep sea sensor networks, where nodes move with currents, it is a costly solution for the shallow lagoon of Venice with

minimal water movement. As previously mentioned, each aMussel robot is able to dive up to the surface of the lagoon and use their ultra long range communication capabilities to alert the base station and therefore, obviates beacon nodes as in Debont et al. (2012) or "data aggregator" nodes. In addition, the construction of an elaborate routing tree as suggested by Zhou et al. (2015) and Debont et al. (2012) necessitates an increased communication payload which is not necessary for event detection in systems such as the robotic swarms developed within the framework of subCULTron.

6.3 The algorithm

To enable a swarm to detect and report an event in an energy efficient manner, I introduce three modes of operation of the robots (or agents): "observation mode", "alert mode" and "event mode". A schematic representation of the different modes can be found in Figure 6.1.

- Observation mode: Initially, all agents are in this mode where they periodically take measurements but refrain from any means of active communication.
- Alert mode: If an agent in the observation mode receives any active communication signal, it will enter the alert mode. In the alert mode an agent also increases its frequency of measurement in order to detect a prospective event as early as possible but refrains from any active communication.
- Event mode: When an agent in observation or alert mode deducts the potential occurrence of an event from its own measurements, it enters the event mode. In this mode, an agent takes measurements with higher frequency in order to observe changes in the environment and collect data with higher temporal resolution. Furthermore, in the event mode agents periodically send 1-bit signals which are received by neighboring agents. If an agent in the event mode receives a signal, it simply relays the signal. In the event mode the agents also estimate the number of other swarm members which are in the event mode. As soon as an agent reaches a sufficiently high estimate of other agents detecting an event, it reports the event to the base station. The mechanism used by the agents to estimate the number of agents in the event mode is explained below.

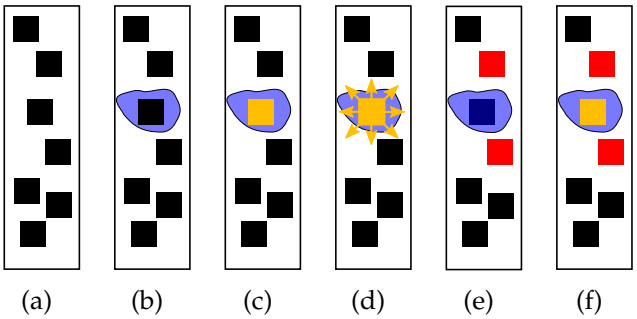
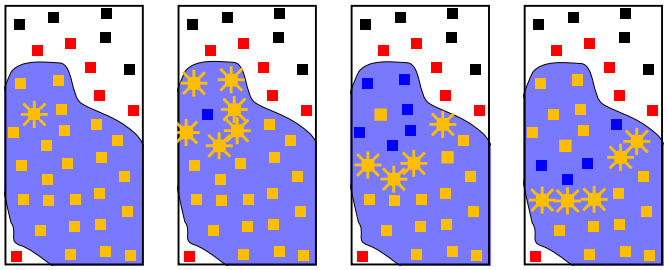


Figure 6.1: A schematic representation of the modes of operation. (a): a swarm of agents in observation mode (black). (b): an irregularity in environmental parameters occurs (blue area). (c): the agent in the blue area takes a measurement and detects the irregularity. The agent enters the event mode (yellow). (d): The agent in the blue area sends a signal to neighboring agents. (e) The neighboring agent receives the signal and subsequently enter alert mode (red). The agent which sent a signal stays insensitive to incoming signals (dark blue) for a defined duration after which, in (f) it is able to receive signals again.

The algorithm is designed to validate the event with a sufficient number of neighboring agents in order to cope with sensor failure and hence, potentially erroneously reported events. The swarm validates the occurrence by estimating the total number of agents in the swarm which are in event mode. Evidently, the concepts introduced in Section 3.2, such as refractory time (t_{ref}), fixed cycle length (t_p^{max}) applies to the aMussel swarm for event detection. Every time an agent in event mode sends a signal, it gets relayed by all the other agents in event mode. This leads to the wave-like propagation of signals through a sub-swarm of agents in event mode which is similar to that demonstrated in Section 3.2. The propagation of a wave through the sub-swarm of robots which has detected an event is schematically shown in Figure 6.2. Once the agent enters the event mode, it executes the WOSP primitive "estimate number of swarm members" in order to estimate how many agents around it are in the event mode. As soon as an agent detects a predefined number of agents (n_*) in the event mode, it dives up to the water surface and reports the event to a base station via the ultra long range communication mode.

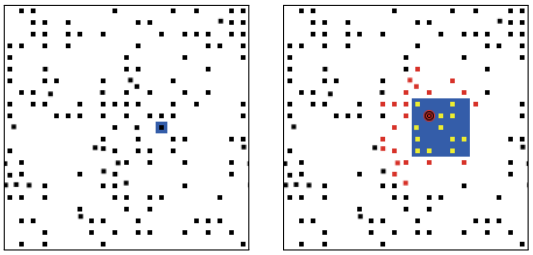


(a) $t = 0$ (b) $t = 1$ (c) $t = 2$ (d) $t = 3$

Figure 6.2: A schematic representation of the relaying of signals in the event mode. (a): An agent in event mode initiates the sending of a signal (yellow star). After signaling, an agent stays insensitive to incoming signals for a specified duration (shown in dark blue). The consecutive relaying of this signal by neighboring agents in event mode is shown in (b) - (d), respectively.

6.3.1 Simulation

Netlogo 6.0.2 is used to conduct simulations. The fundamental units for space and time are referred to as "patches" and "s" respectively. Agents are randomly distributed within a system of size 110×110 patches with periodic boundary conditions. Agents have a communication range of 9, that is, a signal can be received by all agents within euclidean patch-to-patch distance of $r = 9$. The perception of communication for the individual agents is taken to be circular and therefore, the communication area to be $s_A \approx \pi \cdot 81$. At a random position in the system, the anoxia phenomenon is initialized with an area of 1 patch and spreads to all adjacent patches within the Moore neighborhood at a constant rate (s). The agents can detect anoxia solely at the exact position where they are located. All agents choose random times during each of their internal periods at which they measure and potentially send signals. The refractory time during which agents stay insensitive to incoming signals after sending a signal is $t_{ref} = 5$ s. Figure 6.3 shows two screen shots of an exemplary simulation in an early state as well as in its final state where an agent reports the occurrence of an event. If not stated otherwise, the number of agents necessary to agree on the occurrence of an event in order to report it is set to $n_* = 5$. The parameters are deliberately selected to demonstrate the working of the algorithm, and those parameters which affect the performance will be introduced in the upcoming section.



(a) (b)

Figure 6.3: Screen shots of a simulation run of the presented algorithm. The agents are placed randomly in the arena with uniform probability; in (a) and (b) the blue domain represents the presence of anoxia. The black, yellow and red squares represent the individual robots which are in observation mode, event mode and alert mode, respectively. The red target symbol in (b) represents the agent in event mode which sends an alert to the base station.

6.3.2 Swarm level parameters

In order to quantify the performance of the algorithm, I introduce some parameters which reflect the characteristics and performance of the swarm of robots monitoring the environment for the event.

- Measurement and communication periodicity (t_p^{max}): This parameter signifies the periodicity with which all agents in the system communicate. t_p^{max} is measured in unit tick, s.
- Density of robots (D): This parameter measures how densely or sparsely the agents are spread in the environment. D is measured in robots/r^2 where $r^2 \cdot \pi$ is the area of perception of one robot.
- Time until reporting (T): Assuming the start of anoxia at tick $= 0$, the number of simulation ticks taken until an agent sends a message to the base station. The unit of measurement of this parameter is s.
- Area of spread of anoxia(A): The total area anoxia covers until an agent sends a message to the base station. The unit of measurement of this parameter is in patches.

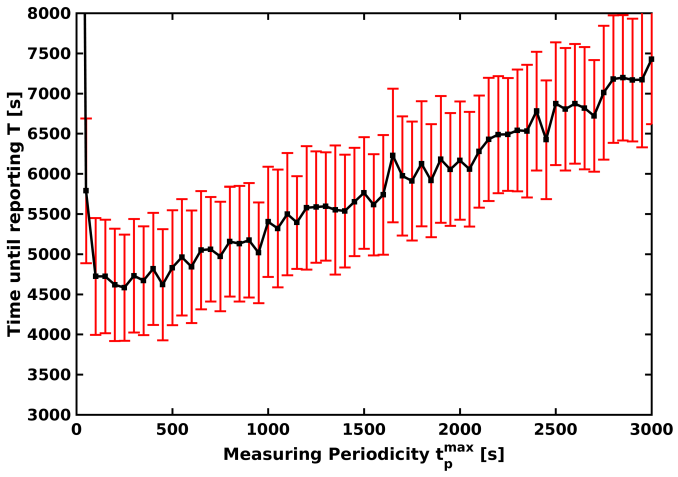


Figure 6.4: Time T until reporting an event versus measuring periodicity t_p^{max}, that is, the periodicity of agents taking measurements of environmental parameters and optionally (if they are in event state) send signals. Parameters: $D = 1$, $t_{ref} = 5$ s.

6.4 Results

In Figure 6.4 the dependence of time T until agents report an event on the periodicity t_p^{max} of agents is shown. Every data point is averaged over 100 independent simulations.

As t_p^{max} approaches 1, the time until reporting diverges towards infinity. For $t_p^{max} < 100$, the agents communicate so frequently that due to the refractory time associated with each broadcast, they rarely receive signals. Thus, agents rarely get to confirm that other agents share their opinion on the occurrence of an event. For $t_p^{max} \approx 300$, the time until reporting has a minimum value of $T \approx 4500\ s$. Thereafter, T grows in an approximately linear manner for increasing t_p^{max}. In the extreme case of $t_p^{max} \to \infty$, the area of the event spreads throughout the system and is detected only when agents first measure and then signal. From Figure 6.4, the optimal measuring periodicity of $t_p^{max} \in [200, 500]$ for which time T is at a minimum is derived. Since within the interval the time to report the event T remains rather stable, $t_p^{max} = 500$ is chosen in order to minimize signal collisions among the pinging agents.

Figure 6.5 shows the time until reporting of an event (averaged over 100

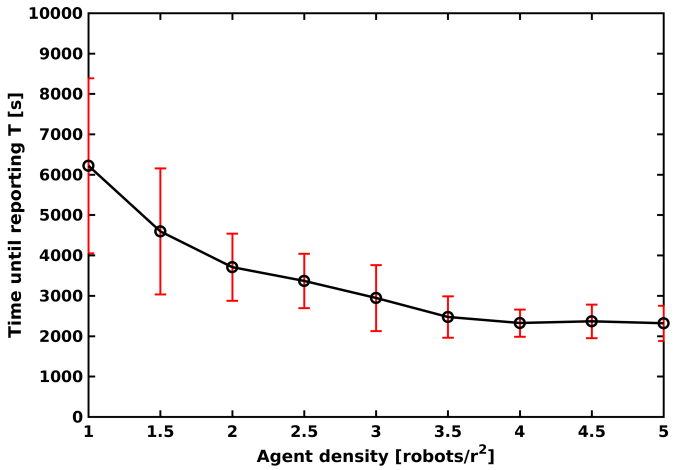


Figure 6.5: Time T until reporting of an event versus agent density D, that is, the number of agents within an area of r^2.

independent simulations per data point) versus the spatial density of agents D, that is, the average number of agents within an area of r^2. For $D = 1$, the average time until reporting of an event is $T \approx 6200\ s$. For increasing D, T decreases until $D = 3.5$ and it subsequently reaches a plateau at $T \approx 2000\ s$. As D approaches 0, T is expected to diverge, since the density is too low for agents to communicate with neighbors and therefore too low to confirm the occurrence of an event. However, for $D > 3.5$, agents are sufficiently well connected such that a further increase in density does not change the collective behavior or performance of the swarm anymore.

Since for $D > 3.5$ the time T does not dramatically decrease any further and therefore the performance in detecting events as fast as possible does not further increase, for the following simulations $D = 3.5$ is chosen as the optimal parameter value.

For a set of agents within the vicinity of an occurring event a set of parameters for optimal performance of the swarm is identified, that is, to minimize the time until reporting an event. However, in case a swarm succeeds in detecting and reporting an event relatively quickly, the area of the event is comparably small and therefore the majority of all agents in the swarm is not within the neighborhood of the event. Furthermore, in a practical application a swarm is likely to be deployed over a long time period until an event might occur. In order to minimize energy consumption not only the

communication between agents can be minimized but also the frequency of taking measurements in the observation mode. In the following, I examined how well a swarm performs if there is a decrease in periodicity t_p^{max} of taking measurements for all agents in observation mode. As soon as an agent enters alert mode or event mode, they adjust their periodicity t_p^{max} to the original value again. Therefore, agents in observation mode measures less frequently, but as soon as they detect an anomaly, they take measurements (and potentially send signals) more frequently.

Figure 6.6 shows the time T until reporting an event versus the factor k by which the periodicity in taking measurements for agents in observation mode is reduced (black circles). All data points shown are averaged over 100 independent simulations. For $k = 1$, agents have the same periodicity t_p^{max} in all states and take on average $T \approx 2700\ s$ until reporting an event. Up to $k = 4$ the time T fluctuates around $T = 2900\ s$ or slightly increases. For $k > 4$ time T increases linearly. For k approaching ∞, agents in observation mode take measurements (linearly) increasingly rarely such that over time the event area spreads out until agents first measure and subsequently report the event. Therefore, for large value of k, a linear increase is expected. The number of measurements taken until reporting of the event is also shown in Figure 6.6. The blue squares denote the total number of measurements taken by the swarm, averaged over the simulation runs. With increasing k, the agents which are in observation mode detect the phenomenon later, thus letting anoxia spread over a larger area. A large number of agents then transitions into event mode, therefore increasing the number of messages sent. The blue crosses denote the total number of measurements taken averaged over each simulation run. As expected, as periodicity is scaled down by an increasing k, the number of measurements taken decreases.

It can be concluded from the graph that for the given system a value of $k = 4$ will produce relatively fast reporting of events while reducing the number of measurements taken and therefore reducing the energy consumed.

6.4.1 Robotic experiments

In order to validate the algorithm, I implemented it on the aMussel robots and tested it under laboratory conditions. Five aMussels were arranged in a linear manner in an arena as shown in the photographs in Figure 6.7

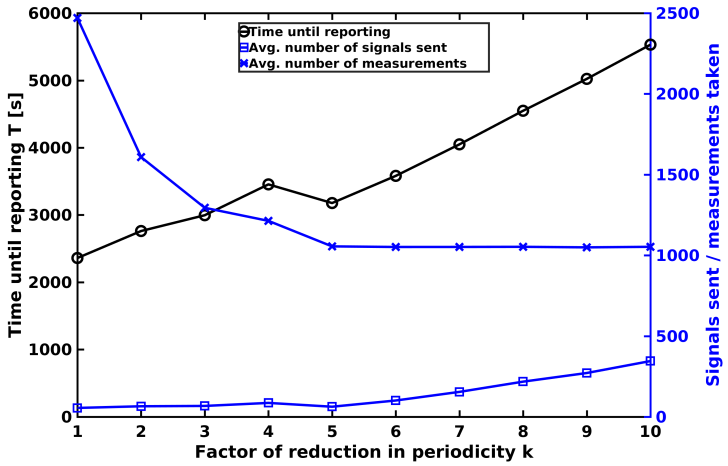
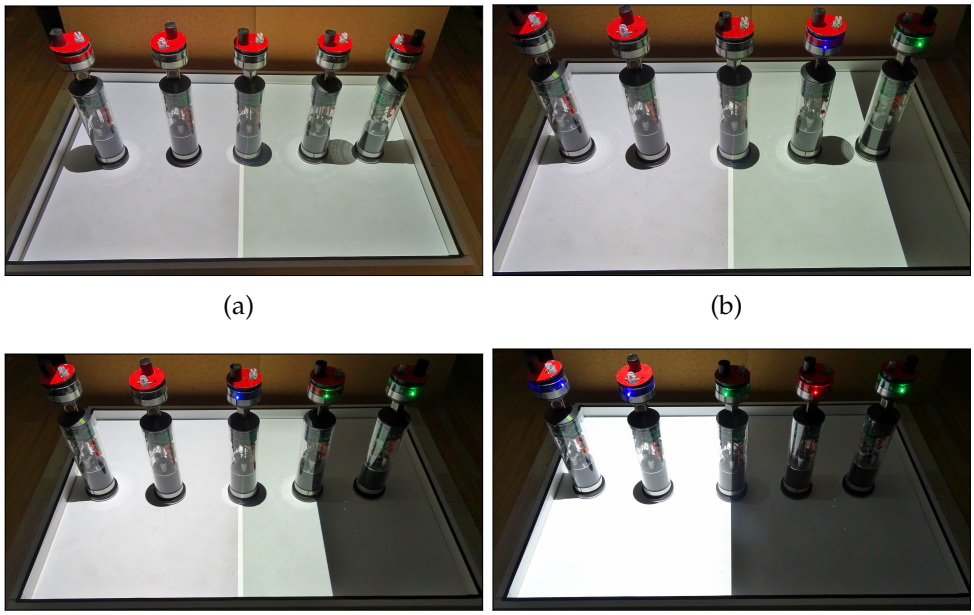


Figure 6.6: Time until reporting T (black circles) versus the factor by which the agents in the observation mode reduce their t_p^{max} as compared to the agents in alert mode or event mode. The number of signals sent and measurements taken (blue squares and crosses, respectively) are also visualized against the factor of reduction k. Parameters: $t_p^{max} = 500$, $t_{ref} = 5$ s.

(a). A projector was used to manipulate the local ambient light around the aMussels to represent oxygen content in the lagoon of Venice. The robots were programmed to register an event when the measured ambient light fell below a particular threshold. As the experiment progressed, the dark patch on the right side of the arena expanded towards the center of the arena as shown in figures 6.7 (b), (c) and (d). The robots were programmed to light up the LEDs on their top-caps to represent their mode of operation. As shown in Figure 6.7 (b), the first robot from the right transitioned into the event mode as represented by the green LED on its top-cap. At the same time, a signal was broadcast by that robot using the short range modulated blue light communication module. That signal triggered the second robot from the right into the alert mode as shown by the blue LEDs on its top-cap. As the dark patch expanded towards the center of the arena (figures 6.7 (b) - (d)), the robots transitioned into the alert mode and subsequently into the event mode. In the event mode, the robots counted the incoming signals to estimate the number of other robots in event mode. In Figure 6.7 (d), the event threshold of $n_* = 3$ was reached for the second robot from the right. It sent a signal via bluetooth to the monitoring station and lit up the red LEDs on its top-cap to signal a validated event in its locality.



(a) (b)

(c) (d)

Figure 6.7: An exemplary run of the robotic experiments conducted with aMussels under laboratory conditions. The agents indicate their modes of operation using the LEDs on their respective top-caps. The blue, green and red LEDs signify the alert mode, the event mode and contacting base station respectively. When there is no LED lit, it means that the agents are in the observation mode.

6.5 Discussion

In the above sections, I presented a simple but practical bio-inspired event detection algorithm for detecting the anoxia phenomenon in the lagoon of Venice using the aMussel robots. Our solution is designed to minimize the amount of communication needed while specifically taking into account the communication and locomotion capabilities of the aMussel robots. Although energy minimization is an ongoing subject of research in sensor networks, the main body of existing literature focuses on reducing the frequency of communication. In contrast to the energy minimization discussed in Zhou et al. (2015) and Debont et al. (2012), this work does not only focus on reducing the number of messages required for effective coordination but also on minimizing the length of each message. From figures 6.4 and 6.5, it can be concluded that the periodicity of measurement and communication as well as the density of distribution of robots are crucial factors in limiting the spreading time of a phenomenon like anoxia. Figure 6.6 implicitly shows the energy consumed for taking measurements by displaying the total number of measurements taken by the swarm as well as the average number of measurements taken per robot.

For a spreading phenomenon it is intuitive that an increase in the number of robots which are monitoring the area ensures a faster detection of a spreading phenomenon. Figure 6.5 shows how the density of robots in the area of interest affects the performance of the algorithm. In order to ensure an interconnected network, a requirement for the presented algorithm to reliably work, each robot needs to be connected to the rest of the swarm through at least one other robot being in its communication range. It follows that there must be at least one robot per sensor area. Assuming a perfectly circular sensor radius of robots, the limits of density of robots D and its relation to sensor area (s_A), and the event threshold (n_*) can be modeled by equations 6.1 and 6.2. In simulations, robots are randomly placed in the environment. In reality, rather than using a naïve placement or random distribution of robots, a set of GPS positions will be generated to deploy the aMussels. Since there is a set threshold of n_* agents necessary to agree on detecting the anoxia across a preset maximum area of A_{max}, the density can be so chosen to be D according to Equation 6.2.

$$D_{min} = 1 \text{ robots}/s_A \approx \frac{0.32}{r^2} \text{ robots} \tag{6.1}$$

$$D = \frac{n_*}{A_{max}} \text{ robots}/s_A \tag{6.2}$$

Figure 6.4 shows the relation between periodicity t_p^{max} of measurement and communication and the time until reporting T. In general, as t_p^{max} decreases, the time taken for reporting the event decreases, since faster measurement and communication enables faster detection. However, below a particular minimum value of t_p^{max} the event cannot be detected due to signal collisions as shown by the initial spike in Figure 6.4. Therefore, t_p^{max} has to be selected so as to minimize the probability of signal collisions. Generally, t_p^{max} has to be large enough to allow each agent to communicate in a different "time slot". Each of these slots consists of a temporal span for each robot to send the single-bit signal and complete its associated refractory time. As a rule of thumb, I used the relation $t_p^{max} \geq 3\ n_*(t_{ref} + 1)\ s$ in order to allow n_* agents to communicate in different temporal slots. The maximum value of t_p^{max} can be selected to minimize T and is therefore a design choice.

Some assumptions were made in this chapter which need revisiting when considering the application of the algorithm to detect anoxia or any other environmental phenomenon. Anoxia is a local phenomenon which spreads to its surroundings. In this chapter, I simplified the dynamics of this phenomenon by assuming a constant spreading rate, starting from a random patch and spreading radially outwards. In reality, the periodicity of measurement and communication as well as sensor placement need to be modified according to the actual dynamics of the phenomenon being examined. In contrast to anoxia, not all events that are of interest have a spreading nature. While tragedies like oil spills move through the water body (Oudhuis and Tengblad, 2018), an aircraft crash (Kaiser, 2014) is an example of an event of interest that does not spread. In this chapter, I assume that an event can be detected easily using simple thresholding of some parameters. While this is true for the detection of anoxia, other methods such as machine learning (Bahrepour et al., 2009) or other event specific techniques can be employed for the detection of events. Even in such cases, the algorithm presented in this chapter can be employed for confirming the detected event with neighbors using minimal communication.

As part of future work, the algorithm presented here will be tested in the field for anoxia detection. There are many more energy saving techniques which can be employed but which are likely to require more than single-bit communication. A study of energy consumed per bit can be conducted in

order to establish the relation between energy consumed and the communication payload. This will enable the implementation of more sophisticated algorithms for event detection. Such a method might allow for a certain amount of node-to-node data exchange.

In conclusion, I presented a simple, bio-inspired, energy conserving event detection algorithm for the detection of the anoxia phenomenon in the lagoon of Venice using aMussel robots. The suggested algorithm can go beyond detecting anoxia using the subCULTron system. Robotic or sensory systems with limited local communication can utilize the algorithm presented here to generate an alarm based on the number of swarm members that detected a local event.

7 Conclusion & Future Work

7.1 Conclusion

Answering the questions raised in Section 1.2 of Chapter 1, I presented a minimalistic design paradigm for swarms. In Chapter 2, I looked at various existing swarm intelligence algorithms and the individual self-organizing solutions they propose to perform commonly found collective behaviors. It was concluded that there is very little work which attempts a design level unification of a variety of behaviors. In Chapter 3, I presented how designing agents in the swarm according to the WOSP paradigm enables the swarm to inherit rich collective behaviors. The paradigm consists of an agent based model for agent-to-agent communication and responding to incoming communication. In a nutshell, WOSP can be explained with a quick example on aggregation. If every agent in a swarm periodically pings and every agent which receives a ping broadcasts it, then traveling waves like "scroll waves" found in slime mold propagate through the swarm. If agents can perceive the direction from which the pings arrive and move towards the pings, they will aggregate at the average origin of pings. Apart from aggregation, quite a few other collective behaviors can be inherited on swarm level by reacting differently to incoming signals. Using the WOSP paradigm, a collective is able to perform leader election, synchronization, localizing an object, estimating the number of swarm members, estimating swarm center, aggregation etc. More importantly, there exists a multitude of possibilities apart from the presented primitives which can be designed using the characteristics of incoming 1-bit ping signals such as incoming ping direction, ping frequency and ping timing. In Section 3.4, I also demonstrated serial combination of individual primitives to perform exploration of an arena and a basic form of collective transport.

Taking advantage of the fact that WOSP aggregation enables a group of agents to aggregate at the area of origin of pings, I designed an emergent

gradient taxis in Chapter 4. Emergent gradient taxis was demonstrated in simulation and was then used as a case to study the resilience of WOSP based behaviors in Chapter 5. It was found through the study that due to the relaying of all pings received, WOSP can stand up to 70% probability of individual failures in agent-to-agent communication. WOSP communication behavior was then used to modify an existing taxis algorithm which initially had low resilience to agent-to-agent communication failures due to the polling mechanism it employed. By plugging in the WOSP communication mechanism of relaying pings, the resilience of the swarmtaxis algorithm to communication failures improved by 30%.

Towards the end of this thesis (Chapter 6), I demonstrated an application of a primitive of WOSP in an underwater robotic swarm deployed in the lagoon of Venice. An algorithm based on WOSP was designed to enable simple robots with limited sensing and locomotion to detect a spreading event with purely local sensing and communication.

7.2 Limitations

In this section the main limitations of WOSP are briefly discussed.

7.2.1 Absolute requriement of connectivity and directionality

As mentioned in Chapter 3, most primitives in WOSP rely heavily on connectivity to the rest of the swarm and the ability to perceive incoming communication in a directional manner. Most sophisticated wireless communication mechanisms which are popularly employed typically are based on radio waves and are not typically directional. Although, such a non-directional technology could be combined with positioning systems or additional reception hardware for directionality, the process is tedious. Such a constraint of connectivity and directionality is not surprising for a bio-inspired paradigm since most swarms in nature evolved in close vicinity and used local signaling mechanisms to coordinate collective actions. Therefore, the WOSP paradigm can be most effectively combined with short range explicit or implicit directional communication for example

based on visual, olfactory or auditory cues. In robotics, light modulated communication, vision based detection, sound based signaling etc. are directional communication techniques that can be used to implement WOSP behaviors.

7.2.2 Scalability limited by communication speed

As mentioned in Chapter 3, since WOSP relies on the relaying of pings, the end-to-end signaling time increases with increasing size and spread of the swarm. Using present technology (such as modulated light communication) there is a substantial time delay for the signal to be passed on from one robot to the other. With the increase in this delay, the time taken to complete the execution of certain primitives (such as "estimating the number of swarm members") increases. Although this is not the case for all primitives, for certain WOSP primitives scalability remains limited by communication speed.

7.2.3 Resilience at the cost of communication

As mentioned in Chapter 5, WOSP shows higher resilience than polling based communication mechanism. It should however be noted that this communication comes at a higher cost of relaying every ping. Although one could argue that single pings used in the case of WOSP will incur lower cost, the cost of communication needs to be weighed against the required resilience while selecting WOSP for an application.

7.2.4 Unification through WOSP

WOSP paves way for a general model of swarm behavior in various natural systems at an algorithm level as envisioned by Sumpter (2005). While such an all unifying model of collective behaviors is still elusive, WOSP might be the start of a discussion which could help us to better understand what unifies swarms in general.

7.3 Future work

Since there is always something one can do with an idea, I would like to point out a few avenues which are worth exploring and researching in the future.

7.3.1 1-bit to multi-bit communication

While I presented a design paradigm for swarms based on single-bit communication, it naturally follows that the use of multi-bit communication would enable more possibilities. One main addition to the existing results would be to implement parallel execution of primitives based on the received pings. In such a case, a longer multi-bit communication would enable certain "layers" of WOSP, with each having a different effect on the agents. With such a setup it would be possible to realize a new level of complexity and enable the usage of the full potential of wave based "programming". Such possibilities are research goals that would add value to the existing paradigm.

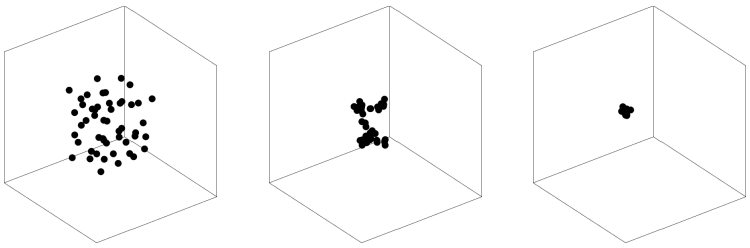
7.3.2 Developing syntax and grammar

As mentioned in Section 3.7, developing a programming syntax and a robotic embodiment for WOSP would be a promising direction worth pursuing in the future. This thesis focuses on presenting a unifying paradigm for several self-organized behaviors. In the future, a WOSP language will be defined with syntax and grammar in order to enable prospective users to combine primitives in a convenient manner and apply them as control scheme to a swarm of robots. Alternatively, existing programming languages like Buzz (Pinciroli et al., 2016) can be used for implementing the WOSP. Availability of a practical implementation of WOSP will further facilitate the usage and application of WOSP and the development of increasingly elaborate primitives allowing easier designing of fully autonomous swarms with the ability to flexibly adapt to varying environmental conditions. Apart from autonomous operation, a programming syntax also paves way for WOSP to be used as a Human-Swarm-Interface. WOSP is a suitable framework for a human swarm interface as the parameters of WOSP can be used

by a human observer to change the behavior of the swarm without needing any underlying design change. In combination with a human observer who can observe the swarm, WOSP primitives can be effectively combined in contrast to the suggestions in Section 3.4, where execution time is used as a basis for estimating transition time from one primitive to the other.

7.3.3 WOSP in three dimensions

Another promising direction of investigation is how WOSP would work in three dimensions. This is especially relevant for applications in underwater robotics or aerial robotics where the agents can typically move in three dimensions. Although a detailed discussion about WOSP in three dimensions is beyond the scope of this thesis, in principle, WOSP should be able to function in three dimensions. If the agents are able to communicate in three dimensions directionally, then the functionalities of WOSP can be preserved in three dimensions. Communicating directionally in three dimensions will involve placing communication transmitters and receptors on different planes on the agent in contrast to the modulated light modules placed on a single plane in the robots in subCULTron (Thenius et al., 2016). In order to illustrate WOSP in three dimensions, a simulation is presented where a swarm executes the aggregation primitive in three dimensions as shown in Figure 7.1. Here, it is assumed that the agents can communicate in three dimensions.



(a) Time: $0.5\,t_p^{max}$ (b) Time: $5.5\,t_p^{max}$ (c) Time: $13\,t_p^{max}$

Figure 7.1: Illustration of a three dimensional adaptation of the aggregation primitive. Initially, the agents are placed randomly in a sphere as shown in (a). Subsequently, the agents aggregate as shown in (b), (c) and (d). In order for primitives to work in three dimensions, it is necessary for agents to communicate in three dimensions. Parameters: $N = 100$, $R_s = 3\,r$, $t_{ref} = 5\,s$, $t_p^{max} = 50\,s$.

Appendix

A List of Publications

A.1 Published

1. Varughese, J. C., Thenius, R., Wotawa, F., and Schmickl, T. (2016). FSTaxis algorithm: Bio-inspired emergent gradient taxis. In *Proceedings of the 15th International Conference on the Synthesis and Simulation of Living Systems*. MIT Press
    - Prof. Thomas Schmickl came up with the idea of using wave propagation in swarm robotics in several early papers. Dr. Ronald Thenius proposed the initial idea for this paper. I implemented, analyzed, authored and presented the paper. Dr. Ronald Thenius, Prof. Thomas Schmickl and Prof. Franz Wotawa were involved in proof reading and giving suggestions for improvement.
2. Thenius, R., Varughese, J. C., Moser, D., and Schmickl, T. (2018). WOSPP - a wave oriented swarm programming paradigm. *IFAC-PapersOnLine*, 51(2):379 – 384
    - Prof. Thomas Schmickl came up with the idea of using wave propagation in swarm robotics in several early papers. Dr. Ronald Thenius proposed the idea of a using wave based communication in swarms for producing multiple behaviors. He proposed the experiments and wrote this paper as an initial concept paper. I helped him with the making of a some figures, proof reading the paper and suggesting improvements based on the reviewer comments. Other authors proof read the paper.
3. Varughese, J. C., Moser, D., Thenius, R., Wotawa, F., and Schmickl, T. (2019b). *swarmFSTaxis: Borrowing a Swarm Communication Mechanism from Fireflies and Slime Mold*, pages 213–222. Springer International Publishing, Cham
    - Prof. Thomas Schmickl came up with the idea of using wave propagation in swarm robotics in several early papers. I further

developed this idea through this paper by implementing the algorithm and writing this paper. Dr. Ronald Thenius, Prof. Thomas Schmickl and Prof. Franz Wotawa were involved in proof reading and giving suggestions for improvement.

4. Varughese, J. C., Thenius, R., Schmickl, T., and Wotawa, F. (2017). Quantification and analysis of the resilience of two swarm intelligent algorithms. In *GCAI 2017. 3rd Global Conference on Artificial Intelligence*, volume 50 of *EPiC Series in Computing*, pages 148–161. EasyChair

   - Prof. Wotawa suggested testing the resilience of the paradigm. I developed this idea, formulated tests, implemented the algorithm and wrote this paper. Dr. Ronald Thenius, Prof. Thomas Schmickl and Prof. Franz Wotawa were involved in proof reading and giving suggestions for improvement.

5. Varughese, J. C., Hornischer, H., Thenius, R., Wotawa, F., and Schmickl, T. (2019a). Collective event detection using bio-inspired minimalistic communication in a swarm of underwater robots. Number 31, pages 634–641

   - Prof. Thomas Schmickl came up with the idea of using wave propagation in swarm robotics in several early papers. I developed this concept further by proposing and developing the idea for this particular paper, implementing the algorithm in both simulation and on robots and by writing the paper. Hannes Hornischer did a part of the analysis of the paper. Dr. Ronald Thenius, Prof. Thomas Schmickl and Prof. Franz Wotawa were involved in proof reading and giving suggestions for improvement.

A.2 Submitted

1. Varughese, J. C., Hornischer, H., Thenius, R., Zahadat, P., Wotawa, F., and Schmickl, T. (2018a). Controlling swarms: A programming paradigm with minimalistic communication. *CoRR*, abs/1804.04202

   - Dr. Ronald Thenius proposed the idea of using wave based communication for a wider application in swarm robotics an earlier paper (mentioned above). This paper was conceptualized, arranged, developed and written by me. I conducted simulations,

analyzed the data, constructed robots, conducted real world experiments and wrote text for most of the sections of this paper. Mr. Hannes Hornischer contributed to parts of the paper where he generated data for graphs and wrote text for some sections. Prof. Thomas Schmickl, Prof. Franz Wotawa and Dr. Payam Zahadat were involved in proof reading and giving suggestions for improvement.

Bibliography

Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight Jr, T. F., Nagpal, R., Rauch, E., Sussman, G. J., and Weiss, R. (2000). Amorphous computing. *Communications of the ACM*, 43(5):74–82.

Ahl, V. and Allen, T. F. (1996). *Hierarchy theory: a vision, vocabulary, and epistemology*. Columbia University Press.

Alcantara, F. and Monk, M. (1974). Signal propagation during aggregation in the slime mould *dictyostelium discoideum*. *Microbiology*, 85(2):321–334.

Alkilabi, M. H. M., Narayan, A., and Tuci, E. (2017). Cooperative object transport with a swarm of e-puck robots: Robustness and scalability of evolved collective strategies. *Swarm Intelligence*, 11(3):185–209.

Amé, J.-M., Halloy, J., Rivault, C., Detrain, C., and Deneubourg, J. L. (2006). Collegial decision making based on social amplification leads to optimal group formation. *Proceedings of the National Academy of Sciences*, 103(15):5835–5840.

Angerer, A., Vistein, M., Hoffmann, A., Reif, W., Krebs, F., and Schönheits, M. (2015). Towards multi-functional robot-based automation systems. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 02, pages 438–443.

Bahgeçi, E. and Şahin, E. (2005). Evolving aggregation behaviors for swarm robotic systems: A systematic case study. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 333–340. IEEE.

Bahrepour, M., Meratnia, N., and Havinga, P. J. (2009). Use of AI techniques for residential fire detection in wireless sensor networks. In *AIAI Workshops*, pages 311–321.

Bak, P., Tang, C., and Wiesenfeld, K. (1988). Self-organized criticality. *Physical Review A*, 38(1):364–374.

Baldassarre, G., Nolfi, S., and Parisi, D. (2003). Evolving mobile robots able to display collective behaviors. *Artificial Life*, 9(3):255–267.

Beckers, R., Deneubourg, J.-L., Goss, S., and Pasteels, J. M. (1990). Collective decision making through food recruitment. *Insectes sociaux*, 37(3):258–267.

Ben-Shahar, O., Dolev, S., Dolgin, A., and Segal, M. (2014). Direction election in flocking swarms. *Ad Hoc Networks*, 12:250–258.

Beni, G. (2005). From swarm intelligence to swarm robotics. In *Swarm Robotics - SAB 2004 International Workshop*, volume 3342 of *LNCS*, pages 1–9. Springer-Verlag.

Beni, G. and Wang, J. (1989). Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?*, volume 102 of *NATO ASI Series (Series F: Computer and Systems Sciences)*, pages 703–712. Springer.

Bjerknes, J. D., Winfield, A., and Melhuish, C. (2007). An analysis of emergent taxis in a wireless connected swarm of mobile robots. In *IEEE Swarm Intelligence Symposium*, pages 45–52. IEEE Press.

Bjerknes, J. D. and Winfield, A. F. (2013). On fault tolerance and scalability of swarm robotic systems. In *Distributed autonomous robotic systems*, pages 431–444. Springer.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.

Bonabeau, E. W. and Theraulaz, G. (1994). Why do we need artificial life? *Artificial Life*, 1(3):303–325.

Bonner, J. T. (1949). The social amoebae. *Scientific American*, 180(6):44–47.

Brambilla, M., Pinciroli, C., Birattari, M., and Dorigo, M. (2009). A reliable distributed algorithm for group size estimation with minimal communication requirements. In *2009 International Conference on Advanced Robotics*, pages 1–6.

Brock, V. E. and Riffenburgh, R. H. (1960). Fish schooling: A possible factor in reducing predation. *ICES Journal of Marine Science*, 25(3):307–317.

Buck, J. and Buck, E. (1966). Biology of synchronous flashing of fireflies. *Nature*, 211:562–564.

Buck, J. and Buck, E. (1968). Mechanism of rhythmic synchronous flashing of fireflies: Fireflies of southeast asia may use anticipatory time-measuring in synchronizing their flashing. *Science*, 159(3821):1319–1327.

Bugrim, A., Fontanilla, R., Eutenier, B. B., Keizer, J., and Nuccitelli, R. (2003). Sperm initiate a Ca2+ wave in frog eggs that is more similar to Ca2+ waves initiated by ip3 than by Ca2+. *Biophysical journal*, 84(3):1580–1590.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2001). Synchronized flashing among fireflies. pages 143–166. Princeton University Press.

Campo, A., Nouyan, S., Birattari, M., Groß, R., and Dorigo, M. (2006). Negotiation of goal direction for cooperative transport. In *Ant Colony Optimization and Swarm Intelligence*, pages 191–202. Springer.

Cavagna, A., Cimarelli, A., Giardina, I., Parisi, G., Santagati, R., Stefanini, F., and Viale, M. (2010). Scale-free correlations in starling flocks. *Proceedings of the National Academy of Sciences*, 107(26):11865–11870.

Cazangi, R. R., VonZuben, F., and Figueiredo, M. F. (2005). BeeAdHoc: An energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2005*, pages 121–128.

Čejková, J. (2015). Aggregation of slime mold *Dictyostelium discoideum*. Youtube Video.

Chisholm, R. L. and Firtel, R. A. (2004). Insights into morphogenesis from a simple developmental system. *Nature reviews Molecular cell biology*, 5(7):531–541.

Christensen, A. L., O'grady, R., and Dorigo, M. (2007). Morphology control in a multirobot system. *IEEE Robotics Automation Magazine*, 14(4):18–25.

Codling, E., Pitchford, J., and Simpson, S. (2007). Group navigation and the "many-wrongs principle" in models of animal movement. *Ecology*, 88(7):1864–1870.

Coore, D. (2004). Towards a universal language for amorphous computing. In *International Conference on Complex Systems (ICCS2004)*.

Couzin, I. D., Krause, J., Franks, N. R., and Levin, S. A. (2005). Effective leadership and decision-making in animal groups on the move. *Nature*, 433(7025):513.

Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In *Lecture Notes in Computer Science*, volume 3342 of *LNCS*, pages 10–20, Berlin, Germany. Springer-Verlag.

Daido, H. (1988). Lower critical dimension for populations of oscillators with randomly distributed frequencies: A renormalization-group analysis. *Physical Review Letters*, 61:231–234.

de Oliveira, D. R., Parpinelli, R. S., and Lopes, H. S. (2011). Bioluminescent swarm optimization algorithm. In *Evolutionary Algorithms*, chapter 5. IntechOpen.

Debont, M., Jamshaid, K., Shihada, B., and Ho, P. (2012). Event localization in underwater wireless sensor networks using monitoring courses. In *2012 1st IEEE International Conference on Communications in China (ICCC)*, pages 769–774.

Decugniere, A., Poulain, B., Campo, A., Pinciroli, C., Tartini, B., Osée, M., Dorigo, M., and Birattari, M. (2008). Enhancing the cooperative transport of multiple objects. In *International Conference on Ant Colony Optimization and Swarm Intelligence*, pages 307–314. Springer.

Devreotes, P. (1989). *Dictyostelium discoideum*: A model system for cell-cell interactions in development. *Science*, 245(4922):1054–1058.

Dorigo, M., Trianni, V., Şahin, E., Groß, R., Labella, T. H., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D., and Gambardella, L. M. (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17(2):223–245.

Ducatelle, F., Di Caro, G. A., Förster, A., Bonani, M., Dorigo, M., Magnenat, S., Mondada, F., Pinciroli, C., Rétornaz, P., Trianni, V., Gambardella, L. M., Ducatelle, F., Di Caro, G., Förster, A., Gambardella, L., Bonani, M., Magnenat, S., Mondada, F., Rétornaz, P., Dorigo, M., Pinciroli, C., and Trianni, V. (2014). Cooperative navigation in robotic swarms. *Swarm Intell*, 8:1–33.

Durston, A. (1973). *Dictyostelium discoideum* aggregation fields as excitable media. *Journal of theoretical biology*, 42(3):483–504.

Durston, A. (2013). *Dictyostelium*: The mathematician's organism. *Current genomics*, 14(6):355–360.

Eberhart, R. C., Shi, Y., and Kennedy, J. (2001). *Swarm intelligence*. Elsevier.

Enright, J. T. (1980). Temporal precision in circadian systems: A reliable neuronal clock from unreliable components? *Science*, 209(4464):1542–1545.

ePuck (2009). e-puck desktop mobile robot - website. http://www.e-puck.org/.

Ermentrout, G. B. (1985). Synchronization in a pool of mutually coupled oscillators with random frequencies. *Journal of Mathematical Biology*, 22(1):1–9.

Ferrante, E., Turgut, A. E., Huepe, C., Stranieri, A., Pinciroli, C., and Dorigo, M. (2012). Self-organized flocking with a mobile robot swarm: A novel motion control method. *Adaptive Behavior*, 20(6):460–477.

Fink, J., Hsieh, M. A., and Kumar, V. (2008). Multi-robot manipulation via caging in environments with obstacles. In *2008 IEEE International Conference on Robotics and Automation*, pages 1471–1476. IEEE.

Garnier, S., Gautrais, J., Asadpour, M., Jost, C., and Theraulaz, G. (2009). Self-organized aggregation triggers collective decision making in a group of cockroach-like robots. *Adaptive Behavior*, 17(2):109–133.

Garnier, S., Jost, C., Gautrais, J., Asadpour, M., Caprari, G., Jeanson, R., Grimal, A., and Theraulaz, G. (2008). The embodiment of cockroach aggregation behavior in a group of micro-robots. *Artificial life*, 14(4):387–408.

Grodzicki, P. and Caputa, M. (2005). Social versus individual behaviour: A comparative approach to thermal behaviour of the honeybee (*Apis mellifera* L.) and the american cockroach (*Periplaneta americana* L.). *Journal of Insect Physiology*, 51(3):315 – 322.

Groß, R., Tuci, E., Dorigo, M., Bonani, M., and Mondada, F. (2006). Object transport by modular robots that self-assemble. In *Proceedings 2006 IEEE*

International Conference on Robotics and Automation, 2006. ICRA 2006., pages 2558–2564. IEEE.

Groß, R. and Dorigo, M. (2009). Towards group transport by swarms of robots. *International Journal of Bio-Inspired Computation*, 1(1-2):1–13.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

Gutiérrez, A., Campo, A., Monasterio-Huelin, F., Magdalena, L., and Dorigo, M. (2010). Collective decision-making based on social odometry. *Neural Computing and Applications*, 19(6):807–823.

Hall, S., Wardle, C., and MacLennan, D. (1986). Predator evasion in a fish school: Test of a model for the fountain effect. *Marine biology*, 91(1):143–148.

Hayes, A. T., Martinoli, A., and Goodman, R. M. (2003). Swarm robotic odor localization: Off-line optimization and validation with real robots. *Robotica*, 21(4):427–441.

Hoff, N. R., Sagoff, A., Wood, R. J., and Nagpal, R. (2010). Two foraging algorithms for robot swarms using only local communication. In *2010 IEEE International Conference on Robotics and Biomimetics*, pages 123–130.

Hölldobler, B. and Wilson, E. O. (1978). The multiple recruitment systems of the African weaver ant *Oecophylla longinoda* (Latreille)(Hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 3(1):19–60.

Huth, A. and Wissel, C. (1992). The simulation of the movement of fish schools. *Journal of theoretical biology*, 156(3):365–385.

Izhikevich, E. M. (1999). Weakly pulse-coupled oscillators, FM interactions, synchronization, and oscillatory associative memory. *IEEE Transactions on Neural Networks*, 10(3):508–526.

Kaiser, S. A. (2014). Legal considerations about the missing Malaysia airlines flight MH 370. *Air and Space Law*, 39(4):235–244.

Karpov, V. and Karpova, I. (2015). Leader election algorithms for static swarms. *Biologically Inspired Cognitive Architectures*, 12:54–64.

Kauffman, S. A. (1993). *The Origins of Order: Self-organization and Selection in Evolution*. The Origins of Order: Self Organization and Selection in Evolution. Oxford University Press.

Kaur, N. and Sood, S. K. (2017). An energy-efficient architecture for the Internet of Things (IoT). *IEEE Systems Journal*, 11(2):796–805.

Kendeigh, S. C. (1961). *Animal ecology*. London: Prentice-Hall International.

Kengyel, D., Schmickl, T., Hamann, H., Thenius, R., and Crailsheim, K. (2011). Embodiment of honeybee's thermotaxis in a mobile robot swarm. In *10th European Conference on Artificial Life (ECAL'09)*, volume 5777/5778 of *LNCS*. Springer-Verlag.

Kengyel, D., Zahadat, P., Kunzfeld, T., and Schmickl, T. (2016). 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 (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*. IEEE Press.

Kennedy, J. S. and Wigglesworth, V. B. (1951). The migration of the desert locust (*Schistocerca gregaria* forsk.) I. The behaviour of swarms. II. A theory of long-range migrations. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 235(625):163–290.

Kim, L. H. and Follmer, S. (2017). UbiSwarm: Ubiquitous robotic interfaces and investigation of abstract motion as a display. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):66.

Koh, H. and Magee, C. L. (2006). A functional approach for studying technological progress: Application to information technology. *Technological Forecasting and Social Change*, 73(9):1061–1083.

Kube, C. R. and Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and autonomous systems*, 30(1-2):85–101.

Kuramoto, Y. (1975). Self-entrainment of a population of coupled non-linear oscillators. In *International symposium on mathematical problems in theoretical physics*, pages 420–422. Springer.

Labella, T. H., Dorigo, M., and Deneubourg, J.-L. (2006). Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(1):4–25.

Le Goc, M., Kim, L. H., Parsaei, A., Fekete, J.-D., Dragicevic, P., and Follmer, S. (2016). Zooids: Building blocks for swarm user interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 97–109. ACM.

Maes, P., Mataric, M. J., Meyer, J., Pollack, J., and Wilson, S. W. (1996). *Robotic "Food" Chains: Externalization of State and Program for Minimal-Agent Foraging*. MIT Press.

Melhuish, C., Holland, O., and Hoddell, S. (1999). Convoying: Using chorusing to form travelling groups of minimal agents. *Robotics and Autonomous Systems*, 28(2-3):207–216.

Michaels, D. C., Matyas, E. P., and Jalife, J. (1987). Mechanisms of sinoatrial pacemaker synchronization: A new hypothesis. *Circulation Research*, 61(5):704–714.

Middleton, E. J. and Latty, T. (2016). Resilience in social insect infrastructure systems. *Journal of The Royal Society Interface*, 13(116):20151022.

Mirollo, R. E. and Strogatz, S. H. (1990). Synchronization of pulse-coupled biological oscillators. *SIAM Journal on Applied Mathematics*, 50(6):1645–1662.

Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I. W., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L. M., and Dorigo, M. (2004). Swarm-bot: A new distributed robotic concept. *Autonomous robots*, 17(2-3):193–221.

Nagpal, R. (2002). Programmable self-assembly using biologically-inspired multiagent control. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 418–425. ACM.

Nagpal, R. (2006). Engineering amorphous systems, using global-to-local compilation. In *Complex Engineered Systems*, pages 291–306. Springer.

Nagpal, R., Kondacs, A., and Chang, C. (2003). Programming methodology for biologically-inspired self-assembling systems. In *AAAI Spring Symposium on Computational Synthesis*, pages 173–180.

Nicolis, G. (1977). Self-organization in nonequilibrium systems. *Dissipative Structures to Order through Fluctuations*, pages 339–426.

O'Keeffe, K. P., Hong, H., and Strogatz, S. H. (2017). Oscillators that sync and swarm. *Nature Communications*, 8(1):1504.

Oudhuis, M. and Tengblad, S. (2018). BP and deepwater horizon: A catastrophe from a resilience perspective. In *The Resilience Framework*, pages 71–87. Springer.

Perez-Diaz, F., Trenkwalder, S. M., Zillmer, R., and Groß, R. (2018). Emergence and inhibition of synchronization in robot swarms. In *Distributed Autonomous Robotic Systems*, pages 475–486. Springer.

Pi, R. (2017). Raspberry pi hardware. `https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md`.

Pinciroli, C., Lee-Brown, A., and Beltrame, G. (2016). Buzz: An extensible programming language for heterogeneous swarm robotics. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3794–3800.

Rabb, G. B., Woolpy, J. H., and Ginsburg, B. E. (1967). Social relationships in a group of captive wolves. *American zoologist*, 7(2):305–311.

Reid, C. R. and Latty, T. (2016). Collective behaviour and swarm intelligence in slime moulds. *FEMS Microbiology Reviews*, 40(6):798–806.

Reynolds, C. W. (1987). Flocks, herds, and schools. *Computer Graphics*, 21(4):25–34.

Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In *Game developers conference*, volume 1999, pages 763–782.

Riedo, F., Chevalier, M., Magnenat, S., and Mondada, F. (2013). Thymio II, a robot that grows wiser with children. In *2013 IEEE workshop on advanced robotics and its social impacts*, pages 187–193. IEEE.

Roberts, G. (1996). Why individual vigilance declines as group size increases. *Animal behaviour*, 51(5):1077–1086.

Ross Ashby, W. (1947). Principles of the self-organizing dynamic system. *Journal of General psychology*, 37:125–128.

Rubenstein, M., Ahler, C., Hoff, N., Cabrera, A., and Nagpal, R. (2014a). Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems*, 62(7):966–975.

Rubenstein, M., Cabrera, A., Werfel, J., Habibi, G., McLurkin, J., and Nagpal, R. (2013). Collective transport of complex objects by simple robots: theory and experiments. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 47–54. International Foundation for Autonomous Agents and Multiagent Systems.

Rubenstein, M., Cornejo, A., and Nagpal, R. (2014b). Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799.

Runca, E., Bernstein, A., Postma, L., and Silvio, G. D. (1996). Control of macroalgae blooms in the lagoon of venice. *Ocean & Coastal Management*, 30(2):235 – 257. Coastal Zone Management in the Mediterranean.

Russell, M. J., Switz, G. M., and Thompson, K. (1980). Olfactory influences on the human menstrual cycle. *Pharmacology Biochemistry and Behavior*, 13(5):737–738.

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

Schmickl, T., Thenius, R., Möslinger, C., Radspieler, G., Kernbach, S., and Crailsheim, K. (2008). Get in touch: Cooperative decision making based on robot-to-robot collisions. *Autonomous Agents and Multi-Agent Systems*, 18(1):133–155.

Seeley, T. D. (1992). The tremble dance of the honey bee: message and meanings. *Behavioral Ecology and Sociobiology*, 31:375–383.

Senanayake, M., Senthooran, I., Barca, J. C., Chung, H., Kamruzzaman, J., and Murshed, M. (2016). Search and tracking algorithms for swarms of robots: A survey. *Robotics and Autonomous Systems*, 75:422–434.

Sherwood, L. (2015). *Human physiology: From cells to systems*. Cengage learning.

Siegert, F. and Weijer, C. J. (1992). Three-dimensional scroll waves organize *Dictyostelium* slugs. *PNAS*, 89(14):6433–6437.

Simons, A. M. (2004). Many wrongs: The advantage of group navigation. *Trends in ecology & evolution*, 19(9):453–455.

Simpson, S. J., McCaffery, A. R., and Haegele, B. F. (1999). A behavioural analysis of phase change in the desert locust. *Biological Reviews*, 74(4):461–480.

Soysal, O. and Şahin, E. (2006). A macroscopic model for self-organized aggregation in swarm robotic systems. In *International Workshop on Swarm Robotics*, pages 27–42. Springer.

Squire, L., Berg, D., Bloom, F. E., Du Lac, S., Ghosh, A., and Spitzer, N. C. (2012). *Fundamental neuroscience*. Academic Press.

subCULTron (2015). Submarine cultures perform long-term robotic exploration of unconventional environmental niches. http://www.subcultron.eu/.

Sugawara, K., Kazama, T., and Watanabe, T. (2004). Foraging behavior of interacting robots with virtual pheromone. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 3074–3079.

Sumpter, D. J. (2005). The principles of collective animal behaviour. *Philosophical transactions of the royal society B: Biological Sciences*, 361(1465):5–22.

Szopek, M., Schmickl, T., Thenius, R., Radspieler, G., and Crailsheim, K. (2013). Dynamics of collective decision making of honeybees in complex temperature fields. *PLOS ONE*, 8(10):e76250.

Tan, Y. and Zheng, Z.-y. (2013). Research Advance in Swarm Robotics. *Defence Technology*, 9(1):18–39.

Thenius, R., Moser, D., Varughese, J. C., Kernbach, S., Kuksin, I., Kernbach, O., Kuksina, E., Miškovi, N., Bogdan, S., Petrovi, T., Babi, A., Boyer, F., Lebastard, V., Bazeille, S., William Ferrari, G., Donati, E., Pelliccia, R., Romano, D., Jansen Van Vuuren, G., Stefanini, C., Morgantin, M., Campo, A., and Schmickl, T. (2016). subCULTron - Cultural Development as a Tool in Underwater Robotics Consortium for coordination of research activities concerning the Venice lagoon system. In *Artificial Life and Intelligent Agents*. Springer.

Thenius, R., Varughese, J. C., Moser, D., and Schmickl, T. (2018). WOSPP - a wave oriented swarm programming paradigm. *IFAC-PapersOnLine*, 51(2):379 – 384.

Trianni, V. and Campo, A. (2015). *Fundamental Collective Behaviors in Swarm Robotics*, pages 1377–1394. Springer.

Trianni, V., Groß, R., Labella, T. H., Şahin, E., and Dorigo, M. (2003). Evolving aggregation behaviors in a swarm of robots. In *European Conference on Artificial Life*, pages 865–874. Springer.

Turgut, A. E., Çelikkanat, H., Gökçe, F., and Şahin, E. (2008). Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2(2-4):97–120.

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

Varughese, J. C., Hornischer, H., Thenius, R., Wotawa, F., and Schmickl, T. (2019a). Collective event detection using bio-inspired minimalistic communication in a swarm of underwater robots. Number 31, pages 634–641.

Varughese, J. C., Hornischer, H., Thenius, R., Zahadat, P., Wotawa, F., and Schmickl, T. (2018a). Controlling swarms: A programming paradigm with minimalistic communication. *CoRR*, abs/1804.04202.

Varughese, J. C., Moser, D., Thenius, R., Wotawa, F., and Schmickl, T. (2019b). *swarmFSTaxis: Borrowing a Swarm Communication Mechanism from Fireflies and Slime Mold*, pages 213–222. Springer International Publishing, Cham.

Varughese, J. C., Thenius, R., Leitgeb, P., Wotawa, F., and Schmickl, T. (2018b). A model for bio-inspired underwater swarm robotic exploration. *IFAC-PapersOnLine*, 51(2):385 – 390.

Varughese, J. C., Thenius, R., Schmickl, T., and Wotawa, F. (2017). Quantification and analysis of the resilience of two swarm intelligent algorithms. In *GCAI 2017. 3rd Global Conference on Artificial Intelligence*, volume 50 of *EPiC Series in Computing*, pages 148–161. EasyChair.

Varughese, J. C., Thenius, R., Wotawa, F., and Schmickl, T. (2016). FSTaxis algorithm: Bio-inspired emergent gradient taxis. In *Proceedings of the 15th International Conference on the Synthesis and Simulation of Living Systems*. MIT Press.

Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 6(75):1226–1229.

Walker, P., Amraii, S. A., Chakraborty, N., Lewis, M., and Sycara, K. (2014). Human control of robot swarms with dynamic leaders. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1108–1113.

Walker, T. J. (1969). Acoustic synchrony: two mechanisms in the snowy tree cricket. *Science*, 166(3907):891–894.

Wang, L., Wang, X., and Hu, X. (2013). Connectivity preserving flocking without velocity measurement. *Asian Journal of Control*, 15(2):521–532.

Webb, B. (1998). Robots, crickets and ants: Models of neural control of chemotaxis and phonotaxis. *Neural Networks*, 11:1479–1496.

Werner-Allen, G., Tewari, G., Patel, A., Welsh, M., and Nagpal, R. (2005). Firefly-inspired sensor network synchronicity with realistic radio effects. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 142–153. ACM.

Winfield, A. F. and Nembrini, J. (2012). Emergent swarm morphology control of wireless networked mobile robots. In *Morphogenetic Engineering*, pages 239–271. Springer.

Winfree, A. T. (1967). Biological rhythms and the behavior of populations of coupled oscillators. *Journal of theoretical biology*, 16(1):15–42.

Wischmann, S., Hülse, M., Knabe, J. F., and Pasemann, F. (2006). Synchronization of internal neural rhythms in multi-robotic systems. *Adaptive Behavior*, 14(2):117–127.

Witkowski, O. and Ikegami, T. (2016). Emergence of swarming behavior: Foraging agents evolve collective motion Based on signaling. *PLOS ONE*, 11(4):e0152756.

Wolfram, S. (1984). Cellular automata as models of complexity. *Nature*, 311(5985):419.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *Stochastic algorithms: foundations and applications*, pages 169–178. Springer.

Zahadat, P., Hahshold, S., Thenius, R., Crailsheim, K., and Schmickl, T. (2015). From honeybees to robots and back: division of labour based on partitioning social inhibition. *Bioinspiration & biomimetics*, 10(6):066005.

Zahadat, P. and Schmickl, T. (2016). Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition. *Adaptive Behavior*, 24(2):87–101.

Zaikin, A. and Zhabotinsky, A. (1970). Concentration wave propagation in two-dimensional liquid-phase self-oscillating system. *Nature*, 225(5232):535.

Zhou, Z., Xing, R., Duan, Y., Zhu, Y., and Xiang, J. (2015). Event coverage detection and event source determination in underwater wireless sensor networks. *Sensors*, 15(12):31620–31643.