Heinrich Heine Universität Düsseldorf

Nature-Inspired Algorithms for Mobile, Communicating, and Sensing Robot Swarms

Inaugural-Dissertation

zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Ahmad Reza Cheraghi

geboren in
Teheran/Iran

Düsseldorf, May 2022

aus dem Institut für Informatik
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Referent: Jun.-Prof. Dr. Kalman Graffi
Korreferent: Prof. Dr. Martin Mauve
Tag der mündlichen Prüfung: 05.07.2022

Abstract

Imagine a world full of robots created to do mundane or dangerous tasks, for example cleaning, carrying heavy objects, protecting us, or searching for survivors in dangerous places. These are examples for the future. But, the complexity of developing these robots should not be underestimated. The hardware and software need to be designed, developed, and tested. Each robot needs to have artificial intelligence. Additionally, the collaboration and communication among robots must work seemlessly.

In this dissertation, we deal with robot swarm, based on three problem statements including 11 research questions. Robot swarms are groups of robots that accomplish tasks that are either impossible to solve by one robot or time-consuming. It is an emerging scientific field.

We design, develop, and evaluate a simulator and seven nature-inspired algorithms for robot swarms. Before doing so, however, we conducted research on robot swarms and summarized 217 publications. The result is an overview of robot swarms' past, present, and future. In addition, a comprehensive review of the current state-of-the-art network simulators has been prepared. This survey compares 25 simulation tools for peer-to-peer, opportunistic, and mobile ad-hoc networks.

We built a new simulator for robot swarms called Swarm-Sim based on the gained insights. Swarm-Sim is easy to understand as it is written in Python. It has an API that is easy to learn, allowing the implementation of a scenario and solutions for various robot swarm tasks. It has an advanced GUI, which displays the animation of the simulation either in 2D or 3D, and allows for changes to the environment ad-hoc. Therefore, Swarm-Sim is a simple to learn and easy to apply simulator to develop and evaluate robot swarm algorithms.

This dissertation's central and essential contribution is the nature-inspired robot swarm algorithms. We implemented and evaluated seven algorithms on the Swarm-Sim and categorized them into three application areas. The first application task is the swarm coating. With swarm coating, the robot swarm is to enclose an arbitrarily shaped object from all sides. We have developed two algorithms for this purpose. Swarm communication is the second application area. We deal with how robots can communicate with each other within the swarm. Both indirect and the direct communication is considered. Last but not least, we deal with swarm movement. We present two algorithms for coordinated movement within the swarm and how to motivate the robot swarm to move in different directions.

As a result, this dissertation provides three modules with ten contribution, presented with summaries from ten papers. The summaries contain two surveys about robot swarms and network simulators, one article about the Swarm-Sim simulator, and seven nature-inspired algorithms for robot swarms. We hope that with this dissertation we have made an important contribution to the scientific community and to the further development of robot swarms.

Zusammenfassung

Stellen Sie sich eine Welt voller Roboter vor, die Arbeiten verrichten, die wir nicht mögen, z. B. putzen, schwere Dinge tragen, uns beschützen oder an gefährlichen Orten nach Überlebenden suchen. Dies sind Beispiele für die Zukunft. Dennoch sollte die Komplexität der Entwicklung dieser Roboter nicht unterschätzt werden. Die Hard- und Software muss entworfen, entwickelt und getestet werden. Jeder Roboter muss über eine künstliche Intelligenz verfügen. Auch die Zusammenarbeit und die Kommunikation untereinander sind wichtige Punkte.

In dieser Dissertation befassen wir uns mit der Kooperation von Robotern, auch als Roboterschwarm bezeichnet, anhand von drei Problemstellungen, mit 11 Forschungsfragen. Roboterschwärme sind Gruppen von Robotern, die Aufgaben bewältigen, die von einem einzelnen Roboter nicht oder nur mit hohem Zeitaufwand gelöst werden können. Hierbei handelt es sich um ein aufstrebendes Wissenschaftsgebiet.

Wir haben einen Simulator und sieben von der Natur inspirierte Algorithmen für Roboterschwärme entworfen, entwickelt und getestet. Zuvor haben wir jedoch Recherchen über Roboterschwärme durchgeführt und fast 217 Veröffentlichungen gelesen und zusammengefasst. Das Ergebnis ist ein Überblick über die Vergangenheit, Gegenwart und Zukunft von Roboterschwärmen. Darüber hinaus wurde eine umfassende Übersicht über den aktuellen Stand der Technik von Netzwerksimulatoren erstellt. Diese Übersicht vergleicht 25 Simulationswerkzeuge für Peer-to-Peer-, opportunistische und mobile ad-hoc Netze.

Aus den Erkenntnissen haben wir einen neuen Simulator für Roboterschwärme namens Swarm-Sim entworfen und entwickelt. Swarm-Sim ist leicht zu erlernen, da es in Python geschrieben ist. Es verfügt über eine leicht zu erlernende API, die die Implementierung eines Szenarios und Lösungen für verschiedene Roboterschwarmaufgaben ermöglicht. Es verfügt über eine fortschrittliche grafische Benutzeroberfläche, die die Animation der Simulation entweder in 2D oder 3D anzeigt und die Möglichkeit bietet, die Umgebung ad-hoc zu verändern. Daher ist Swarm-Sim ein einfach zu erlernender und leicht anzuwendender Simulator zur Entwicklung und Bewertung von Roboterschwarm-Algorithmen.

Der zentrale und wesentliche Teil dieser Dissertation sind die von der Natur inspirierten Roboterschwarm-Algorithmen. Wir haben sieben Algorithmen auf dem Swarm-Sim implementiert und evaluiert und sie in drei Anwendungsbereiche kategorisiert. Die erste Anwendungsaufgabe ist das Schwarm-Coating. Beim Schwarm-Coating soll der Roboterschwarm ein beliebig geformtes Objekt von allen Seiten umschließen. Hierfür haben wir zwei Algorithmen entwickelt. Die Schwarmkommunikation ist das zweite Anwendungsgebiet. Es geht darum, wie Roboter innerhalb des Schwarms miteinander kommunizieren können. Dabei werden sowohl der indirekte als auch der direkte Weg der Kommunikation betrachtet. Zu guter Letzt beschäftigen wir uns mit der Schwarmbewegung beschäftigt. Dafür stellen wir zwei Algorithmen für die koordinierte Bewegung innerhalb des Schwarms vor und zeigen, wie man den Roboterschwarm motivieren kann, sich in verschiedene Richtungen zu bewegen.

Daher bietet diese Dissertation drei Module mit zehn Beiträgen, und alles wird mit zehn Zusammenfassungen von zehn Arbeiten präsentiert. Die Zusammenfassungen enthalten zwei Übersichten über Roboterschwärme und Netzwerksimulatoren, einen Artikel über den Swarm-Sim-Simulator und sieben von der Natur inspirierte Algorithmen für Roboterschwärme. Wir hoffen, dass wir mit dieser Dissertation einen wichtigen Beitrag zur wissenschaftlichen Gemeinschaft und zur weiteren Entwicklung von Roboterschwärmen geleistet haben.

Acknowledgements

Being humble and grateful and believing and trusting in yourself leads to success. Success is reaching your goal no matter how arduous; it is staying focused, being patient and passionate, and believing in your goal.

For more than eight years, I started my PhD journey, and it brought me to my limits. This PhD journey was not only a challenge for my mind , it was a challenge for my entire being. I gained more knowledge than I thought possible. I once thought I was someone who could manage everything with such ease, but through this journey, I have begun to see things more seriously and handle obstacles with more precision. And now this journey is coming to an end. I am now writing my acknowledgment as the last part of my dissertation.

However, this is not the end of my educational journey. I have come to find out that, "I know that I do not know." The ocean of knowledge is endless. Whatever we know is not enough, there is still more knowledge we can obtain, and as the Prophet Muhammad PBUH quotes: "Seek knowledge from the Cradle to the Grave." Now this part of my life is coming to an end, and my next aim is to keep improving on my existing skills and gain more knowledge. I will grow steadily and not to stop. I am honored that I have been able to attain my PhD in my beloved city Duesseldorf, and specially at the Heinrich Heine University.

For that reason, my first gratitude goes to my first supervisor, Dr.-Ing. Kalman Graffi, who gave me the opportunity, trust, and belief to start this PhD journey. His patience and dedicated time are priceless. One of the main reasons I continued with my PhD through the long nights and trying times, was his trust and belief in me. Words cannot express my gratitude, dear Kalman, because what you did for me cannot be expressed with words. Thank you very much for your trust and support through this journey.

Next, I want to thank my second supervisor Prof. Dr. Martin Mauve. You are an inspiration, and your advice is priceless. You gave me the necessary pressure and deadline to publish two papers in one year. Based on these two factors, I could published more than eight papers in one year. Where is a will, there is a path. Dear Martin, thank you very much for everything.

The third part of my gratitude belongs to my beloved bachelor and master students, Abdelrahman Abdelgalil, Karol Actun, Asma Ben Janete, Jochen Peters, Fabio Schlösser Vila, Sahdia Shahzad, Gorden Wunderlich, and Julian Zenz. These students supported me in my research. We worked like a swarm together. We were productive and efficient and the result is that we could produce nine fantastic papers together.

I am also grateful to the entire team of the Computer Networks department and, in particular, to the members of the "Technology of social networks" workgroup for a friendly working environment. My special thanks goes in no particular order, to Dr. Newton Wafula Masinde, Dr. Tobias Amft, Dr. Andreas Disterhöft, Dr. Andre Ippisch, Dr. Raed Al-Aaridhi, Dr. Alexander Schneider, Dr. Christian Meter, and Dr. cand. Ahmad Rabay'a. I miss all the social events and discussions we had. Especially passing the chain of blame to the one who was late for lunch. Moreover, I want to thank my first office mate, Philipp Hagemeister, my second office mate, Dr. Salem Sati, and my last and current office mate, Dr. cand. Raphael

Bialon. Further, I want to thank Sabine Freese for all the administration work, she handles for all of us. And of course your kindness and patience.

Additionally, I would like to thank Dr. Christian Dumpitak and Dr. Debbie Radtke from the iGRAD office. You are doing a great job by providing scientific and practical seminars for us. It was an honor for me to be the student representative at the iGRAD, and thank you so much for that.

Further, I want to thank my parents, Mohammad Ebrahim Cheraghi and Khadijeh Eskandar Afshari, for their prayers, trust, and belief in me. Then, to my younger brother, Milad Cheraghi. We both supported each other in our educational process in these eight years. A special thank you goes out to my grandfather, Mohammad Eskandar Afshari, who has been a role model for me. More thanks go to my entire family, my aunts, uncles, and cousins. Dear family, you all are really important to me, and thank you for always being there to support me. I love you all.

Last but not least, I want to thank myself, frankly speaking. It took more than eight years to finish this dissertation, but I did not quit, and I managed to publish ten papers, based on my determination and patience. I had a lot of tough times and was close to giving up many times. But, I persevered. My aim was to get my PhD, and there was nothing that could stop me. I always said if someone gives me billions of dollars to quit my PhD, I would deny it. Because I started this journey, and I will finish it. So, now I finished the writing of my dissertation, I am close to getting my PhD, the journey is coming to a bittersweet end. Therefore, the last prop goes to me.

Nevertheless, I hope I did not forget anyone, and if so, I ask for forgiveness.

Again thank you all.

May the 4th be with you!

Contents

Chapter 1

Introduction

In the year 2100, the likelihood of robots populating the streets and taking over our daily chores is very high. Therefore, let us imagine that these robots are integrated into our everyday life. They clean the streets, pick up our luggage when we arrive at the airport, or drive our cars. They may act as as soldiers who protect us, artisans who build or renovate our houses, as robot drones that deliver packages, or as rescue teams that help us in a disaster. In addition, nanometer-sized robots could be injected into our bodies to check our health, protect us from viruses, or cure us of diseases. This collaboration of many robots that leads to solving tasks that would be impossible or inefficient for one robot alone is called a robot swarm, and the inspiration comes from nature. In this generation or the next, we may soon be confronted with robot swarms.

In this chapter, we first present the motivation for this dissertation. Next, the problem statements and the research questions necessary for robot swarms are elaborated. Third, we explain the main modules, and their contributions. Finally, this chapter ends with the outline of this dissertation.

1.1 Motivation

A swarm is a collection of many units to solve a task that is either impossible or difficult to solve alone. The world is full of natural wonders, and the source of robot swarms comes from nature. For example, antibodies that envelop and destroy a virus, ants that spread pheromones to help other ants find their way to the food source, or fish that band together to fend off predators. These wonders and their swarm behavior are the sources of inspiration and vision for a decentralized swarm of robots that work together to complete tasks in a time-efficient manner.

The human body is a marvel. It has billions of neurons, and blood flows throughout the body, etc. Most of the organs work independently without the impact of us humans. One of the miracles is the immune system. When a known virus enters the body, the human immune system starts to react and sends antibodies to destroy the virus. They look for the virus and surround it to destroy it. As shown in Fig. 1.1, we can see a Covid-19 virus surrounded by antibodies. This coating of the virus with antibodies is a miracle. However, how are the antibodies programmed to coat a virus cell?

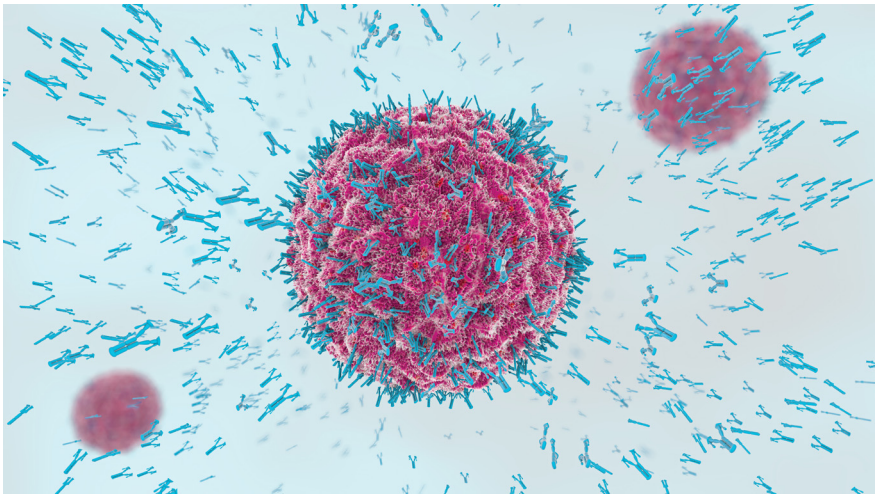


Figure 1.1: Antibodies surrounding a Virus (Source: [1])

Another example is ants searching for food (Fig. 1.2). The ants walk around randomly to find food. However, when they find food, they use indirect communication for guiding other ants to the food source. This indirect communication works by using pheromones. Pheromones are chemical secretions that the ants spread on the way back to its nest. As a result, a trail is created that leads other ants to the founded food source. Ants that encounter this pheromone trail will follow it. Thus, with the help of pheromones, the food foraging becomes more efficient. But how can robot swarms use this indirect communication, and are there other ways of communication? What are the risks of the different forms of communication?

The last natural wonder we will look at is the fish shoal. In a fish shoal, many hundreds of small fish come together and swim together in the sea, as shown in Figure 1.3. With having such as huge formation, the fish can protect themselves from larger predators because the resulting size seems frightening and robust. However, what is fascinating about a fish shoal is that all the fish have the same speed, keep their distance from each other, and simultaneously change their directions. But how do they manage this collective movement without getting in each other's way?

There are many more examples of natural miracles. Nevertheless, everything works without a central control unit. The motivation for this dissertation is to build algorithms that solve tasks with the help of robot swarms. However, before the dissertation is discussed, here is a short overview of necessary words and their definitions.

Swarm Robotics and Robot Swarms The robots developed today mainly work independently rather than as a team. Therefore, it is necessary to develop robots and algorithms to act together as a team. We refer to this collective team of robots as a robot swarm. However, most of the literature uses the term swarm robotics definitions for swarm robotics are as follow:

Figure 1.2: Ants are following a pheromone trail(Source: [2])

- *"Swarm robotics is the study of how to design a large number of relatively simple physically embodied agents in such a way that local interactions between the agents and between the agents and the environment result in a desired collective behavior."* [4]

- *"Swarm robotics can be defined as the study of how a swarm of relatively simple physically embodied agents can be constructed to collectively accomplish tasks beyond the capabilities of a single agent."* [5]

Our definition of swarm robotics serves not only with robots that move but also with robot parts such as the hand or foot. For example, the hand consists of several fingers, which sometimes have to act together as a team to grasp something. On the other hand, a robot swarm we define as a swarm of fully functional robots. Nevertheless, the terms robot swarm and swarm robotics are used with the same meaning in this dissertation with the following definition: *"Robot swarms or swarm robotics is the collaboration of many programmable (consist for example of hardware or biological) subjects acting together as a team to accomplish tasks in a time-saving, practical, and more efficient manner. The swarm size can start as small as two subjects and increase as long as the efficiency does not suffer."*

Location A location in this dissertation is the point for a two or three-dimensional space. An object or subject can be positioned on a location. Therefore, we use location to refer to the point at which an object or subject is positioned. A location has a memory, and therefore its minimum capability is to store data so subjects can read from and write on it.

Figure 1.3: A fish shoal (Source: [3])

Object, tile, item Anything that cannot carryout actions are defined as an object. An object can be carried, dropped, or walked on. It has a memory that allows the subject to store and read data. An object represents stones, staples, soil, etc. They are helpful to build houses, walls, borders or various geometric formations, etc. We refer to the object sometimes as a tile or item because the spectrum of meaning is very versatile.

Subject, agent, particle, robot A subject can perform actions i.e. it can move, pick up or drop items etc. It can be a robot within the swarm, an software agent distributed across multiple computers, a fluid particle, or programmable DNA. As we can see, there are so many possibilities a subject can use. Therefore, in this dissertation, we also refer to the subject as an agent, particle, or robot. There are no definitional differences between them, and all of them act as a leading entity that can solve tasks together as a robot swarm.

All the necessary terms have been defined. We will later introduce how locations, objects and subjects are modeled mathematically. Now that we know the primary motivation and the necessary definitions, it is time to explain the problem statements.

1.2 Problem Statement and Research Questions

There are no problems only challenges. In a challenge you must find a solution to the given situation. Nevertheless, the challenge here is abstracting the swarms, inspired by nature and their environment, into a model and building algorithms for them. This section briefly summarizes the challenges we faced in robot swarms research. It includes three problem statements

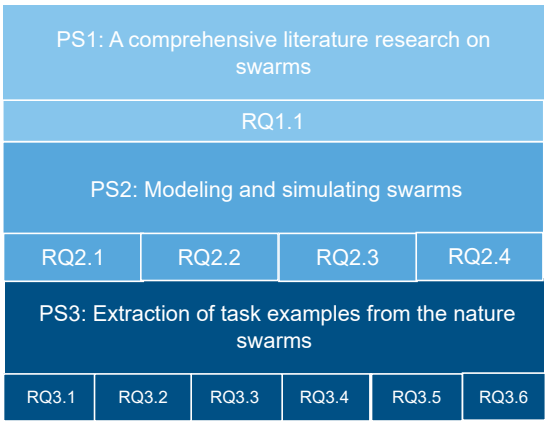


Figure 1.4: Problem Statement Overview

(PS), and each of them consists of one or more research questions (RQ) (See Fig. 1.4). The PS describes the current challenge that needs to be solved, and the RQ consists either of one question or many based on the context we will answer in this dissertation. The first challenge is to gain knowledge about robot swarms. Therefore, we start to do comprehensive literature research. The second challenge is about how to model robot swarms. The question between using real robots or simulators will be dissected; and the challenge of seeking applications and tasks inspired by nature and writing algorithms for them will also be discussed. Nevertheless, let us start with the first problem statement.

PS1: A comprehensive literature research on swarms The world is changing, and digitization is taking place. Many predictions from movies and books have already become a reality or are in the process of becoming one. Nevertheless, we do not know exactly how robot swarms function. Therefore, the first challenge is to examine books and publications to learn about robot swarms.

RQ1.1 What do we know about robot swarms' past, present, and future? This dissertation deals with the following questions: what do we know about the past of robot swarms? Who are the pioneers of robot swarms? What did they do? Are there other types of grouped robots? What is the difference between multiagent systems and robot swarms? Next, we need to examine the current state of robot swarms. What projects have been completed and are still ongoing? Do swarm robots exist? What tasks are they performing? What about simulators for robot swarms? Do they exist, and are they maintained or up to date? Finally, we come to the future of swarm robotics. Based on research, what can we predict about the future of robot swarms? What is our vision of robot swarms?

The theoretical knowledge about robot swarms is one part of this dissertation. However, we have to convert this knowledge into research. Therefore, the next challenge is to find ways of feasibility for robot swarms in science.

PS2: Modeling and simulating swarms We see that animals act collaboratively without any central entity coordinating them from nature. They all act together as a swarm to build a synergy to solve tasks that are impossible or time-consuming for only one. Our second challenge is to bring those inspired swarms into the science world, i.e., model them and their environment mathematically. The following research question is necessary to proceed further.

RQ2.1 Should real or simulated robots be used for this research? We found the answer to this question very quickly. The main goal of this dissertation is to extract applications from the natural swarm and develop algorithms for them. Therefore, we need more than two robots to test robot swarm algorithms. That is, we need to buy robots or simulate them. However, after googling for real robots that can act as a swarm, we came across Kilobots [6]. They are small robots that can act as a swarm. However, the price for them is high. Ten Kilobots cost about 1,700 €. Therefore, buying 100 robots to test robot swarm algorithms is expensive and updating each robot with a new algorithm would be time-consuming. Therefore, the answer to this research question is to use a simulator because it is cost-effective, saves time, and does not incur the overhead of updating each robot. However, finding a suitable simulator for algorithm evaluation is another challenge that leads us to our second research question in this part.

RQ2.2 What is the current state of simulators? For the second research question, it is necessary to know the current state of simulation tools on the market. It is essential to conduct comprehensive research on simulators for the three types of wireless networks, i.e., mobile ad-hoc, opportunistic, and peer-to-peer networks. This research question is necessary to learn more about the current state of simulators. We need to know how many simulators have been developed for these networks. With what programming language are they written? How easy are they to learn? Which simulator can handle all three networks? When was the latest version released? Finally, how popular are they in the research community? These research questions need to be answered before proceeding to the following research question.

RQ2.3 How to model mathematically natural phenomenon? The inspiration for robot swarms comes from natural phenomena, and we want to simulate them and develop algorithms for them. However, it is necessary to model them in mathematical sets to use them for algorithms independent of programming languages. These mathematical models will form an abstraction of the natural swarm. The goal is to use these models to describe the algorithm for each defined task, and to determine when a task is complete and prove that the algorithm works correctly. Therefore, the research question in this part is: How should the natural swarm be modeled mathematically?

RQ2.4 How do we develop a simulator? For this challenge, the research question is: How can a simulator be designed that is that is easy to learn, and not complicated to develop, and able to evaluate new algorithms? What features should simulators have to differentiate themselves from other simulators? With what programming language should it be written? Finally, should it have a graphical user interface with 2D or 3D animation and how should the results be interpreted?

These research questions build a fundamental for this dissertation. Because based on answering those questions, we can proceed further to solve the following upcoming challenges.

PS3: Extraction of task examples from the nature swarms Modeling nature into an abstract mathematical model gives us the input necessary for simulating robot swarms. Nevertheless, it is vital to define tasks, which brings us to our last challenge: "Extracting tasks from the nature swarms". The aim here is to extract tasks from nature, then define their aims, necessary inputs, algorithm, and evaluation. To standardize our proceeding to extract robot swarm tasks and the procedure to simulate them, we have set the following research questions.

RQ3.1 What inspiring tasks can we extract from nature? Based on the Cambridge dictionary, a task is defined as: "A piece of work to be done, especially one done regularly, unwillingly, or with difficulty." Therefore, a task is a piece of work that has to be done. With this research question we can extract tasks inspired by nature. Research must be done on natural phenomena to find a suitable task that can be adapted for robot swarms. A simple example of a task inspired by nature is ants' food collection (foraging). Ants leave their nests to find food. When they find a food source, they take it and return to its nest. Ant foraging is one example of a nature-inspired task. Nevertheless, each task must define its aim, input, solution, and metrics.

RQ3.2 What is the aim of the task? The main question here is, what is the aim? For each task, it is necessary to define its aim. For example, with the ant foraging, the task is to find food and take it to the nest. Therefore, their main aim is to increase food capacity in the nest. Knowing the task and the aim, it is still important to know what inputs are necessary and which characteristics they should have.

RQ3.3 What is given for the task? After we find out the task and the aim, it is necessary to know what is given. What are the required inputs to solve the task with the given aim? Additionally, what capabilities should they have? Who are the actors and what are their features and limitations? Moreover, how should the environment for the robot swarm be set up? For example, the inputs are the ants that can move around, a restricted area, and a given amount of food sources by the ant foraging. To sum it up, the inputs must be defined with their capabilities to design and develop a solution, which brings us to our fourth research question.

RQ3.4 How should the task be solved? After knowing the task and the inputs, the next required step is designing and implementing the algorithms. An algorithm is a step-by-step procedure that tells the swarm robots how to solve the task. For designing the algorithm, either a flow chart or pseudo-code can be used. After designing, it is necessary to implement it into a programming language to test and validate the algorithm, which brings us to the following research question.

RQ3.5 How can the algorithm for the task be validated? Any developed algorithm needs to be validated for its usage, which means that there must be confirmation that it is fulfilling the task and reaching its aim. Therefore, it is necessary to define the criteria for the validation. For example, to validate the ant foraging, we test if the amount of provided food is collected in the nest after a given time. It is crucial to define a timeframe, otherwise, an unsuccessful algorithm can go on for infinity. The time can be defined based on a scenario. Therefore, we know that the ant foraging is working correctly. However, there might be other factors necessary for the validation, but time is also essential.

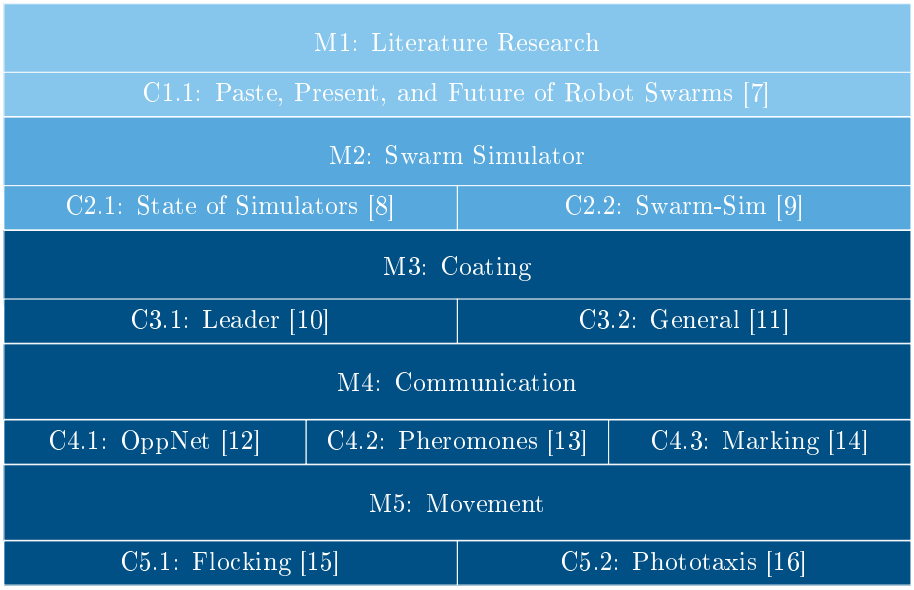


Figure 1.5: Modules Overview

RQ3.6 What are the metrics for the task, and how should they be measured? Now, we must identify how to measure the algorithm and what metrics will be used. Metrics can be success rate, overhead, time, messages, etc. However, the success rate is the most common metric because it is necessary to see if the algorithm is doing the task within a given time and condition. Therefore, the success rate is the most essential metric. For example, the ants' foraging algorithm is successfully validated when all the food is found and taken to the nest within a given time. To sum up, metrics are vital for validating and testing the efficiency of an algorithm.

We now know the three problem statements and each of their research questions. Next, we describe three modules and the contributions of this dissertation.

1.3 Modules and Contributions

Robot swarms are a group of robots doing tasks that are either impossible for one robot or more efficient when they are done together as a swarm. This dissertation has three modules (M) with contributions (C) as shown in Fig. 1.5. The contributions are grouped based on the previous modules. The first contribution is comprehensive literature research about the past, present, and future of robot swarms that consists of a summary of almost 217 publications. Next, we build a state-of-the-art simulator named "Swarm-Sim" that is easy to learn. However, before developing it, we elaborate the state of network simulation tools, and additionally, we model the nature swarm mathematically. The final contribution is the central part of this dissertation. We design, develop, and evaluate seven different algorithms for three practical tasks for robot swarms. The first task is the coating of objects with robot swarms. for robots to surround

an object of any shape from all sides. The communication between the individual robots within the swarm is the second task. In this task, the aim is to find different communication techniques and test them within the swarm. And finally, the last task is to move the swarm in any direction. Because, the swarm is built out of robots, it is necessary to find possibilities to keep them together while moving as a swarm.

M1 Literature research about robot swarms It is always important to begin with reviewing current existing literature. This module extracts all the necessary information from the robot swarm's past, present, and future analyze the exact definitions and challenges. We read and summarize various publications about the past and the present of robot swarms. Therefore, our first contribution is about the world of swarm robotics from the beginning to the future. We contribute an extensive literature survey about the pioneers and future prediction of robot swarms.

C1.1 Past, Present, and Future of Swarm Robotics In chapter 2, we summarize our published extensive study [7] on the past, present, and future of robot swarms, which is our first contribution. In this publication, we have summarized about 217 papers. We start with the past of robot swarms and give a brief overview of the pioneers of robot swarms. We then review the critical approaches and definitions of swarm robotics. In this part, we describe the types of robot swarms, their characteristics, the advantages and problems they have, and their tasks, domains, and different applications. Next, the different robotic systems are discussed. We will describe systems robotics, swarm robotics, multi-robotics, multi-agents, and sensor networks. We define agents as programmable entities such as robots, computers, routers, antibodies, viruses, etc. After describing the pioneers and definitions of robot swarms, we move to the current state.

The current state of robot swarms consists of natural robot swarms and simulators. We first give an overview of the current state of all real robot projects for robot swarms. There are only a handful of robot projects, and most of them are not present, except for Kilobots from Harvard University. The second paper compares all state-of-the-art swarm simulators. We then give an overview of the future of robot swarms and what will happen based on our research and opinion. Knowing the past, present, and future of robot swarms, we decided to build a new simulator, our second significant scientific contribution.

M2 Developing a State-Of-The-Art Simulator for Robot Swarms The second module is to build a robot swarm simulator. Our first contribution to this module is a survey about the different types of simulators for network simulators because all robots in the swarm form a network. Therefore, we examined the most advanced network simulators in preliminary. The second contribution is Swarm-Sim. We present this simulator with its mathematical model of natural swarms abstractly.

C2.1 The State of Simulation Tools In chapter 3.1 we answer the question, what is the current state of network simulators on the market? There are three types of networks suitable for robot swarms: mobile ad-hoc, opportunistic, and peer-to-peer networks.

A mobile ad-hoc network (MANET) allows units to connect instantly. The units are mobile and constantly on the move, and the position of each unit can change, which poses a challenge for routing protocols. The second network is the opportunistic network (OppNet). The OppNet is a delay-tolerant network, which means it can tolerate delays of messages. In typical standard routing protocols, there is always a time-to-live (TTL) counter so that the message sent does not remain in the network forever. However, in OppNet, a TTL cannot function optimally because the connection between the individual units is occasionally established and based on possibilities. Thus, the connection between sender and receiver is not always given. To combat this, each unit in OppNet has to accept a delay. Therefore, new routing protocols that can endure time delays have been created. Last but not least, there is the peer-to-peer (P2P) network. A P2P is an overlay network built on top of a network such as the Internet, MANET, or OppNet. The advantage of P2P is decentralized, meaning that data can be shared within entities without a central entity such as a server. It has many complex protocols for data sharing, such as searching and discovery. Therefore, a combination of these three networks is helpful for robot swarms. For this reason, we decided to investigate the current simulators for these three networks.

In the investigation, we read many publications. We tested some simulators and checked their popularity based on the number of citations provided from the Internet. However, none of the simulators could simulate all three networks simultaneously. Nevertheless, we concluded that only a few could do so. The result of this research is presented in our second paper, "The State of Simulation Tools for P2P Networks on Mobile Ad-Hoc and Opportunistic Networks." [8] published and summarized in chapter 3.1.

C2.2 Swarm-Sim: A 2D & 3D Simulation Core for Swarm Agents Swarm-Sim [9] is a simulator developed by us with inspiration from the Paderborn University [17, 18]. It has an API to write a *Scenario* quickly, project an environment with all the entities, and a *Solution* to write algorithms to solve a task. Additionally, it has a GUI for 2D or 3D animation and changing the *Scenario* by deleting or adding *Matters* and many other features. Swarm-Sim is open-source, and it is entirely written in Python. From the nature inspired we designed a mathematical model as follows.

The world consists of passive and active matters. Real-world examples of passive *Matters* are stones, bricks, walls, buildings, etc. Generally, a passive *Matter* is everything that cannot do any actions on its own. On the other hand, active *Matters* are entities such as the human, animals, the cells or antibodies in our body, robots, computers, or human beings. To wrap it up, whatever is programmable or can do actions on its own are active *Matters* and anything that cannot do any activities is termed passive *Matter*.

We define each *Matter*in this world as a unique entity. It is like DNA or fingerprint, and therefore, it has its unique ID. Therefore, our first mathematical model is a set of unique natural numbers representing all the *Matters*. Let M be the set of elements. Each $m \in M$ is an exact positive natural number representing the ID of

a *Matter*, which brings its uniqueness like the fingerprint of humans.

$$M = \{m \mid m \in \mathbb{N}_+\}$$

Any entity that can do actions we termed here as an *Agent*. An *Agent* A is an active *Matter* which can do activities such as moving, taking, and dropping. Agents are elements of *Matters*. In contrast, passive *Matters* can be manipulated but do not initiate actions. They cannot walk or do any activity. However, they can be carried by *Agents* or the *Agents* can walk on them. We term passive *Matters* as *Items*.

Let $A \subseteq M$ be the set of *Agents* A and let $I \subseteq M$ be the set of *Items* I. Together they form a partition of M: $A \cup I = M$ and $A \cap I = \varnothing$. However, every *Matter* must be located on a ground, where we can identify its position.

A ground where *Matters* can be located is termed *Location*. L be a set of natural numbers which can be mapped with *Matters*, to localize their position.

$$L = \{l \mid l \in \mathbb{N}\}$$

Each *Matter* is mapped to a *Location*. The function is defined as $\lambda : M \to L$ and $\lambda(m) = l$. For animating the actions of *Matters* while simulating, it is necessary to map the *Location* with coordinates to visualize the position changes of the *Matter*. This is not important for the mathematical model, but it is necessary to mention for the visualization in the Swarm-Sim.

Swarm-Sim provides both a 2D and 3D Euclidean Space grid. Each *Location* l is a vector that consists of either of two or three coordinates as follow:

$$l = \begin{cases} (x, y) \in \mathbb{Z} & \text{for} \quad \mathbb{R}^2 \\ \\ (x, y, z) \in \mathbb{Z} & \text{for} \quad \mathbb{R}^3 \end{cases}$$

The task of the robot swarm algorithm is to change the state of each *Matter* until they reach the aim. A state can be a movement, including taking or dropping objects. Therefore, all the *Agents*, *Items*, and *Locations* must have a state. For example, on *Location*, it is possible to read from and write on it, and therefore, its condition is either written or unwritten. Additionally, a *Matter* can be positioned on a *Location*; the state of *Location* changes from free to occupied. For *Items*, the state can be either dragged or undragged, and for *Agents*, it can be, for example, to have taken an *Item* or to be free. Therefore, *Items* and *Agents* have a state, too, which can be changed due to the given situation and algorithm. Therefore, let ST be the vector of all the states.

$$ST = ST_L \times ST_I \times ST_A$$

ST_L, ST_I, ST_A are the set of *Locations*, *Items*, and *Agents*, i.e., written, dragged, or carrying. As a result, the modeling of the swarm consist of *Matters* $L, I, A \in M$ and their states ST.

We now define a general abstract of the robot swarm mathematical model. However, we aim to simulate them on the Swarm-Sim. Therefore, we project the above quantities to a model for the Swarm-Sim.

Swarm-Sim is a round-based simulator for simulating *Matters* as the robot swarms and their environment. Thus, after each round, each *Matter* state can change. Let R be a set of rounds, which $\hat{r}$ is the maximum round, and $r \in R$ is a natural number. As a result:

$$R = \{r \mid r \in \mathbb{N}_0, r \leq \hat{r}\}$$

A *Matter* is positioned on a *Location* in Swarm-Sim. *Matters* combined with *Locations* is termed world W. Which is a set of the mapping of *Location* and *Matter*. The following model for world W is defined as: W^r the set of *Matters* and *Locations* at time $r \in R$ joined.

$$W^r = L^r \cup M^r$$

However, since each *Matter* has its own state *st* and changes while the simulation runs, we define a new set termed Swarm-Sim World SW. It is the mapping of the world with the set ST.

$$SW = (W, ST)$$

The Swarm-Sim-World SW consists of W, i.e., all the *Locations*, mapped with *Agents* or *Items*, and all the states of the *Matters* ST.

Nevertheless, this dissertation aims to build algorithms for solving swarm tasks extracted from nature. Each algorithm programmed in Swarm-Sim is termed *Solution*. A *Solution* is a set of instructions for changing the states of each *Matters* after each round, i.e., ST^r. Therefore, *Solution* SOL is a transition state function that changes the state of *Matters* ST^r to a new state ST^{r+1} after each round.

$$SOL(ST^r) \rightarrow ST^{r+1}$$

It must be mentioned that ST^r is a subset of ST, i.e., $ST^r \subset ST$. Nevertheless, each *Solution* must have an initial starting point.

An initial starting point in Swarm-Sim is termed *Scenario*. A *Scenario* consists of all the *Matters*, i.e. *Agents* or *Items* positioned on a *Location* with a state at round $r = 0$.

$$SCE = \{(W^0, ST^0) \mid ST^0 \in ST, W^0 \in W\}$$

Additionally, the scenario SCE is equal to Swarm-Sim World at round zero SW^0.

$$SCE = SW^0$$

However, a task is successful when it reaches its aim, which means that all the *Matter* states must reach a final state. This defined state is labeled Finale State FST. The FST is *Matters'* final goal. Therefore, the simulation terminates successfully when state ST^r has reached FST, and the *Solution* for the given task is valid. However, when the simulation round time r is equivalent to the maximum round time, $\hat{r}$, the *Solution* could not fulfill the task, which is a failure.

Finally, the Swarm-Sim is a finite-state machine consisting of the following tuples:

$$(SW, SCE, SOL, \hat{r}, FST)$$

First, the Swarm-World SW has the *Matters* and the *Locations*, and their states ST. Next, the *Scenario* SCE indicates the starting position of all the *Matters* at

the beginning of the simulator. Third, all the matters need to perform a task to reach an aim. To fulfill the mission, a solution SCO (algorithm) is given, which needs to be done within the maximum round time $\hat{r}$. Otherwise, the mission is not successful, and the simulator terminates. The simulator completes successfully when the final state FST is reached.

The above mathematical model is a summary of the model from the publication [9]; the outline of this contribution is in chapter 3.2. To summarize, Swarm-Sim is easy to learn. It has an API that gives the necessary interfaces to build a scenario and its solutions. Until now, more than 12 theses with algorithms for various use-cases have been developed on this simulator. Therefore, it brings a considerable contribution to the science world.

The contribution for M2 is the design and development of Swarm-Sim with comprehensive research about network simulators and Swarm-Sim's mathematical models. However, a simulator needs tasks to simulate it, and for this research, we extracted seven tasks for three applications, which is our third main contribution.

M3 Coating Objects with Robot Swarms This contribution uses the help of a robot swarm to coat objects. Imagine there is a cancer cell that needs to be eliminated, and we have nanobots that can be injected into the body to find the cancer cell. When the nanobots reach the cancer, they need to surround the cell to take it out or destroy it from all sides. Therefore, coating algorithms will help to cure diseases in the future. We contribute two coating algorithms as follows.

C3.1 A Leader Based Coating Algorithm for Simple and Cave Shaped Objects The leader-based coating aims to coat an object by dragging and dropping robots by the leader. The leader of the swarm first scans the object to find out the size and type (i.e., either simple- or cave-shaped). By scanning, the leader discovers the position for the coating and the number of robots needed to coat. After the scanning is finished, the leader takes each robot to position them around the object. The leader-based coating algorithm, however, faced many challenges coating caved-shaped objects. The summary of this contribution and the challenges with their solutions can be read in chapter 4.1. The leader-based coating can only coat caved-shaped and specific objects. It cannot cover arbitrarily shaped objects. The following contribution can coat any shaped object.

C3.2 General Coating of Arbitrary Objects Using Robot Swarms The general coating of an arbitrarily shaped object follows a different strategy than leader-based coating. This coating algorithm uses the entire swarm instead of just one leader. When the swarm hits an object, it starts to coat the object. The strategy of general coating is that each robot makes space for the others to get closer to the object. These strategies continue until all the robots surround the object. However, this strategy is complicated because the robots share information about the accessible locations, leading to redundant information within the swarm. The solution to this problem and the exact general coating algorithm description can be read in chapter 4.2.

M4 Challenges and Possibilities of Communication for Robot Swarms Communication within the swarm is vital to share information. This information can be data calculated, assumed, or taken from the environment. Therefore, we decided to research communication types for robot swarms. In this part, we contribute three types of swarm communication. The first one is opportunistic communication through opportunistic networking (OppNet). Hence, the robots are near each other, and a network of robots is built. By OppNet, the swarm shares data only based on opportunistic connection, which means that only necessary data is sent when a connection is established. The first contribution for communication is to make the Swarm-Sim capable for OppNet. We implemented two existing OppNet routing protocols (Epidemic and ProPHET) and tested them. The second and the third contribution we dedicate to stigmergy. Stigmergy is an indirect way of communication. Here the communication is through marking locations or pheromones. This type of communication does not concern the direct communication between the robots, e.g., through WiFi or Bluetooth. Instead, each robot puts information for other robots in the environment. Nevertheless, communication within the robots is a vital task, and therefore, the following three contributions are dedicated to it.

C4.1 Opportunistic Network Behavior in a Swarm: Passing Messages to Destination The opportunistic network (OppNet) is a delay-tolerant network. The traditional network, such as the Internet or MANET, needs small latency for communication, and thus it cannot tolerate delays. Therefore, for OppNet, new routing protocols need to be used that can take delays. We updated the Swarm-Sim with two OppNet routing protocols, the Epidemic and ProPHET. We tested and evaluated them against each other to determine which one is better and suitable for Swarm-Sim and robot swarms. The result of it is shown in chapter 5.1.

C4.2 Prevention of Ant Mills in Pheromone-Based Search Algorithm We present the indirect communication using pheromones in this contribution. Pheromones are fragrances spread by ants to let other ants know that they have found food and to give directions. However, the pheromones vaporize after a time. However, the pheromone communication is faulted; it causes the ants to die. How this happens, the solution for it and the evaluation of the pheromone-based communication can be read in chapter 5.2.

C4.3 Universal 2-Dimensional Arbitrarily Shaped Terrain Marking How do robot swarms know if the terrain is already conquered? In this last contribution of the communication method for robot swarm, we provide a combination of direct and indirect communication. The robots are equipped with wireless communication tools and can additionally mark the location. Here the robots visit unknown and arbitrarily-shaped terrain. By marking the visited terrain locations, the robots indirectly communicate which locations have already been seen. Additionally, they broadcast the position that have been marked. Therefore, other robots are prevented from re-discovering terrain by receiving information or sensing the marked location. Nevertheless, marking the terrain makes it more efficient, too. Thus, we present two types of marking algorithms and an algorithm to avoid obstacles in this contribution. In chapter 5.3 the algorithms and the results are presented.

M5 Dynamic Movement for Robot Swarms Moving as a group of robots is challenging and requires some rules. Therefore, we dedicate this post to the movement of robots. In this contribution, we first introduce the rules for moving as a swarm. We use swarm motions like a group of birds in flight, termed flocking. Flocking has some constraints and limitations that we simulate. The second part is about the effect of light on swarms. We tested if light could affect the direction of a swarm.

C5.1 Robot Swarm Flocking on a 2D Triangular Graph Flocks are a phenomenon that originates from birds or fish. In a swarm, the animals do not interfere with each other. They have the same speed and stay close to each other. We used a three-zone model in this paper and adapted it in Swarm-Sim. This model has three ordered zones based on the distance to the other robots. A robot checks based on his vision and senses in which zone its neighbors are. If the neighbors are in the first zone, they are too close. Therefore, it must slow down. However, if the robots are in the second zone, the distance is optimal. But, if the robots are in the last zone, it must speed up because the probability of getting out of the swarm is high. We tested three-zone models with different zone and swarm sizes. The summary and the results of the contribution can be found in chapter **??**.

C5.2 Phototactic Movement of Battery-Powered and Self-Charging Robot Swarms A phototactic phenomenon allows the swarm to move toward or away from the light without knowing the emission direction. As soon as a being perceives light, it begins to move. This synergistic movement allows the swarm almost to guess the direction of emission. In this paper, we adapt the phototactic behavior into the swarm sim. We use two types of robot swarms. The first type is a battery-powered robot, and the second type is a self-charged robot. The battery-powered robots start moving as soon as they sense light, while the self-charging ones need some time to recharge before they start moving. We pit both types against each other to see which of the two swarm types reaches a given finish line faster. The summary and result of this paper are presented in chapter 6.2.

1.4 Outline

In this chapter, the robot swarms were introduced and the problem statements and research questions were defined. Additionally, we presented the modules and contributions of this dissertation. The following chapters provide summaries of the publications based on the previous contributions.

Chapter 2 gives a summary and elaborate on the past, present and future of the robot swarm. It presents the study written and inspired by more than 217 publications. First, we describe how the robot swarms started in the research and who the pioneers of robot swarms are. Next, we give an overview of the different types of robot swarms. Additionally, the advantages, issues, and application fields are discussed. Followed by a summary of the actual state of robot swarm systems and simulation tools. Lastly, an outcome about the future of robot swarms wraps up at the end of this chapter.

In chapter 3 we guide you to the development phases of a robot swarm simulator called "Swarm-Sim." We first summarize a survey (chapter 3.1) about all the evaluation tools for P2P networks on MANET and OppNet to determine if it is necessary to develop a new simulator. In this survey, we describe all the simulators of those networks and compare them. The second part of this chapter discusses the reasons for building a new simulator. In chapter 3.2, we introduce our simulator Swarm-Sim. We outline the features, the use cases, and how to develop quickly new scenarios and solutions for them. Swarm-Sim is a unique and straightforward simulator that allows to development and evaluation of robot swarm algorithms.

Chapter 4 elaborates the first part of module nature-inspired swarm algorithm module. We first handle the leader-based coating for simple and cave-shaped objects in chapter 4.1. We describe the algorithm, the challenges we faced, and how we solved them. In the second 4.2 we present the general coating algorithm for arbitrarily-shaped objects. In contrast to leader coating, the general coating algorithm can coat any shaped object and use the whole robot swarm without leaders.

In chapter 5, the communication of robot swarms are discussed. We consider two types of communication methods: direct and indirect communication. First, direct communication involves the robot communicating over waves (e.g., WiFi or Bluetooth). We use the opportunistic network for direct communication, and chapter 5.1 is dedicated to it. chapter 5.2 describes the indirect communication. Indirect communication considers spreading information in the environment so the other robots can read unknowingly. The chapter ends with the marking algorithm in chapter 5.3. The marking algorithm combines indirect and direct communication.

Chapter 6 shows the challenges for robot swarm movements. The robots within the swarm must obey rules to move together without interfering. Therefore, in chapter **??** we present the rules of how the swarm should move as a flock. The next chapter 6.2 of this chapter is about how light can affect the robot swarm's directions. The swarm senses light and should move away from it without knowing where it comes from.

Finally, we conclude our contributions and give our deductions on the three modules: Literature Research, Swarm Simulator, and Nature-Inspired Swarm Algorithms and discuss possible future work in chapter 7.

Chapter 2

The World of Robot Swarms from the Beginning to the Future

What do we know about robot swarms? In this chapter we give precise information on robot swarms. We take you to the past of the robot swarms and show you the pioneers of robot swarms and what kind robot swarms exist. From here, we look at the present day. The actual projects of robot swarms are presented, and we take a look at which simulators are on the market. A vision of the future and what kind of use cases can be created end this chapter. At the end, this chapter summarizes the knowledge gained from 217 publications.

2.1 Past, Present, and Future of Swarm Robotics

This section summarizes the contributions and gives a verbatim copy of our paper [7].

Ahmad Reza Cheraghi, Sahdia Shahzad, Kalman Graffi
"Past, Present, and Future of Swarm Robotics"
In: *Proceedings of the Springer 2021 Intelligent Systems Conference (IntelliSys). Volume 3.*
2021.

2.1.1 Paper Summary

This survey aims to give a broad overview of swarm robotics. The knowledge of more than 217 research publications has been summarized in this survey. We are starting with the past of swarm robotics, and we name the pioneers in this research field. Further, we answer questions such as which people came up with the idea and what criteria must be fulfilled for swarm robotics. Additionally, the properties, advantages, and issues of swarm robotics are considered, and we look at how swarm robotics are helpful. In the present part of this survey, we present today's innovations and simulators for swarm robotics. Ending with the future, we describe our visions of swarm robotics, what practical innovations might emerge, in which direction it will go, and what challenges are still open for this exciting and emerging scientific field.

The History of Swarm Robotics

The word swarm robotics was used for the first time by two different researchers, G. Beni [19] and Fukuda [20] in 1988. G. Beni first talked about cellular robotics. These robots are autonomous and act in an n-dimensional cellular space. Additionally, they have no central entity; they have a common aim to solve a task, but limited communication capabilities. Fukuda defined these robots as autonomous human cells that accomplish complex tasks.

In 1993 Gregor Dudek et al. [21] gave swarm robotics features, such as topology, communication range, size, etc. He also mentioned that swarm robotics is a multi-robotic system. Thus, it is unnecessary to have two definitions.

Later, the focus of swarm robotics was on attaining more information on natural swarm phenomena and realizing them, especially in the areas of *foraging, flocking, sorting, stigmergy*, and *cooperation* [22, 23, 24, 25, 26]. However, G. Beni gave with his publication in 2004 [27] a broad and precise definition about swarm robotics. He said that the robots are simple, identical, and self-organized. He defined the size of the swarm to $10^2 - 10^{<<23}$ robots, which means between 100 and much less than 100^{23} robots. Further, he gave some qualification points for swarm robotics, i.e., scalability, robustness, and flexibility. G. Beny had given the most precise definition for swarm robotics, especially by defining the properties of swarm robotics.

Nevertheless, swarm robotics is the science of discovering how robots can act and communicate with each other to reach a desired goal. On the other hand, robot swarms are a group of heterogeneous robots acting together to solve tasks. However, swarm robotics is a scientific field, and there are plenty of approaches.

Swarm Robotics Approaches There are a lot of scientific field approaches for swarm robotics, which we elaborate more precisely:

1. Types of swarm robotics.
2. Properties that swarm robotics should have.
3. The advantages and issues in swarm robotics.
4. Tasks and areas of robot swarm.
5. Different application fields that suit swarm robotics.

Swarm Types Through the years, as the scientists were working on imitating natural swarms to robots, they categorized these swarms into four types.

1. *Biological swarms*: The author [28] uses the term biological swarm for a natural phenomenon where the animals collaborate. This term is used commonly in swarm robotics.
2. *Swarm intelligence*: Intelligence means "to have the ability to learn, understand, and make judgments or have opinions that are based on reason" [29]. Therefore, in a swarm of robots, their intelligence is understood when the robots collectively learn and gain

information. Based on that information, they decide or judge as one entity which is termed swarm intelligence [30].

3. *Swarm behaviors*: When many robots act together to solve tasks it is termed Swarm behaviors. Fish swimming as shoal or the birds flying as a flock are example of swarm behavior; they are either swimming or flying together.

4. *Swarm engineering*: This term was introduced for the first time by [31]. He defined a two-way process of how to design swarm robotics. First, a task with various conditions must be defined for one individual robot to solve. Finally, behaviors must be defined for a group of robots to fulfill their task, and their condition as a swarm should be assessed.

Swarm Properties As mentioned before, G. Beni [27] created with a precise definition of swarm robotics and mentioned the properties for swarm robotics systems as follow:

1. *Scalable*: A swarm robotic system must be able to handle the increase or decrease of robots within the swarm.

2. *Robust*: The robots must adapt themselves to changes within the swarm. That means that if a robot becomes defective, all other robots should not lose their focus to solve the task.

3. *Flexible* Here, the robots must adapt themselves to changes that happen to their environment. If an obstacle appears, all of them should be able to handle it without losing their aim.

Further, some additional properties are helpful and commonly used for other multi-agent and sensor systems. Those properties are:

1. *Autonomy*: Robots that act on their own without any central unit to control them.

2. *Self-organized*: Here, the robots should be capable of organizing themselves when changes are happening.

3. *Self-assembly*: Automatically organizing themselves, without any external help, into patterns or shapes.

4. *Decentralized*: Having a centralized unit for swarm robots is difficult when it comes to scalability, flexibility, and robustness. Because, in all three properties, all the robots must be acting together to accomplish those properties. As a result, decentralization must be given between the robots to avoid a single point of failure.

5. *Stigmergy*: Stigmergy is the indirect communication between the robots. The idea of Stigmergy comes from the animal world. For example, ants spread pheromones in the environment so other ants can find their way to a food source. This indirection communication is helpful for the swarm robotics system.

Swarm Advantages and Issues There are many advantages with swarm robotics, but naturally, there are some disadvantages as well. Based on [32] the advantages of swarm robotics are that they are autonomous and can adapt themselves to changes within their environment. Further, being a swarm makes the robots more powerful compared to being alone. Additionally, swarm robotics systems are flexible. Therefore, they can be used in different fields or categories. They can solve the problems no matter how big or small the swarm is. A swarm robotics system is much faster because of parallelism. Tasks are divided into subtasks that are solved by each robot within the swarm. Lastly, the robots are simply made, therefore, cost-efficient. Nevertheless, there are some disadvantages as well, which are mentioned in [23]. Swarm robotics systems issues are only interesting for a small portion of applications because it is decentralized. Due to automation, the robots may act suddenly differently than they are supposed to do. It is hard to develop them into the actual live application with a 100 percent success rate. Finally, each robot must have global knowledge in real life application, which is difficult to realize.

Task areas and Tasks for Swarm Robotic Systems: Task areas and Tasks for Swarm Robotic Systems: Robots are primarily used in areas, which are dangerous or time-consuming for humans. Based on literature reviews [33, 34, 35, 36], there are many swarm robotics areas. For example, **tasks in specific regions**, which are areas only made for the specific type of swarm robotic system, such as collecting trash in parks. Next, **tasks in dangerous zones** where the swarm is doing tasks that are dangerous for humankind, such as going into a burning house to find victims. Another example is **tasks in changeable areas**. Here are circumstances that can change (e.g., the weather in a natural disaster). Therefore, the swarm must be flexible and scalable to adapt itself to changes in this area. Another area for swarm robotics systems is **redundancy areas**. In these areas, the robots within the swarm should adapt themselves to the loss of robots. An example of such an area can be war zones, where robots are getting killed by enemies. Now that we know the areas for swarm robotic systems, it is necessary to describe some tasks.

1. *Forming shapes and patterns:* The task here is to create a specified shape (e.g., stars, statues, etc.). Many studies regarding these tasks can be read in [37, 38, 39, 40].

2. *Aggregation:* Aggregation in swarm robotics means grouping different robot swarms together to solve a task. More can be read in [41, 42, 43].

3. *Coordinated movements:* This task is inspired by the animals such as fish shoals, where the fish swim together in a coordinated way. The same task is for swarm robotics. All the robots should be capable of moving together coordinated in any given direction. Research regarding coordinated movements are presented in [44, 45].

4. *Distribution of robots to cover area:* In this task, the robots are separated to monitor their environment more efficiently. Therefore, it is the opposite of aggregation. The author in [46] introduces an algorithm for such a task.

5. *Searching for specific sources:* Searching for sources is an emerged task for swarm robotics. In this task, the robots can spread to find sources. However, it is necessary to find sources in less time and let other robots know about the founding. One example of food searching in a probabilistic way can be read in [47].

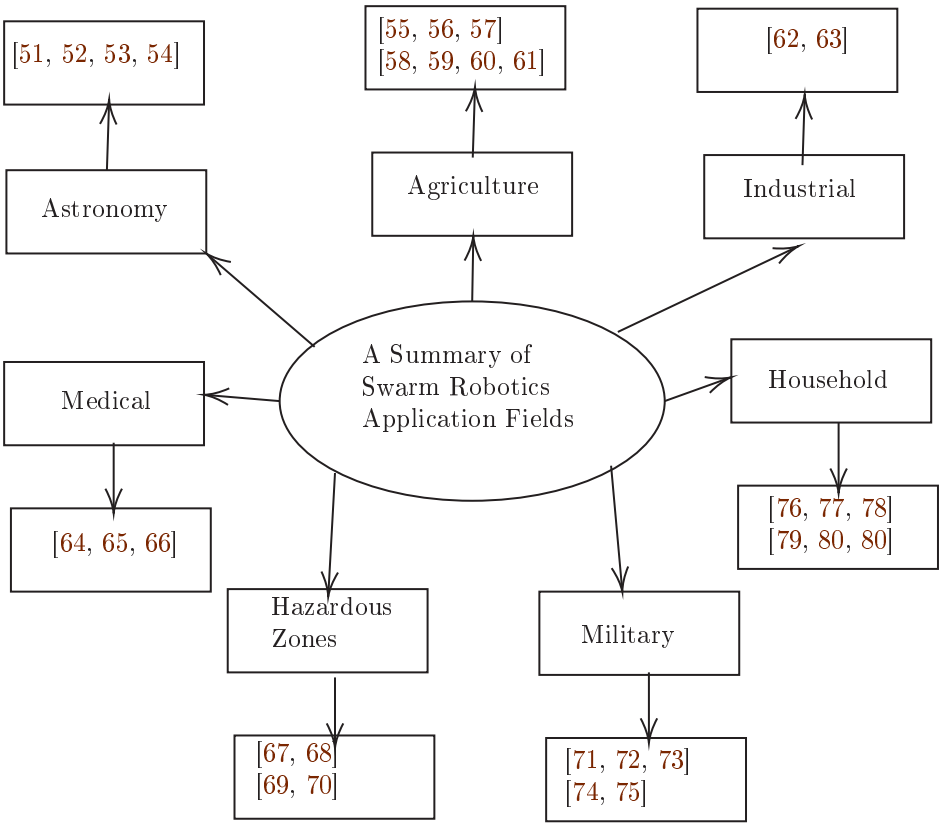


Figure 2.1: A summary of swarm robotics application fields [7].

6. *Scanning an area and Navigation:* Scanning an area and sharing the data to another robot to navigate there easily is another task. An example can be the robots scanning a fire zone to find humans, and consequently navigating to that area for rescuing. More can be read in [48, 49]. *Transporting objects:* Here the robots transport objects that cannot be transported alone, like ants transporting objects more significant than their size. In [50] there are examples of transporting objects with a swarm.

We mentioned the task areas and tasks that are suitable for swarm robotics. Next, we present the application fields of swarm robotics, i.e., real-life cases they can be applied.

Application Fields Swarm robotics is currently used in many application fields. An overview of the application fields is shown in Fig. 2.1.

1. *Agriculture:* Working as a famer, such as the monitoring of the field or seeding; swarm robotics also have the ability of working more efficiently.

2. *Industrial:* In industry, time is very important. Therefore, using the robot swarm for distributed industrial works, such as assembling cars, can be time-saving and good for the industry.

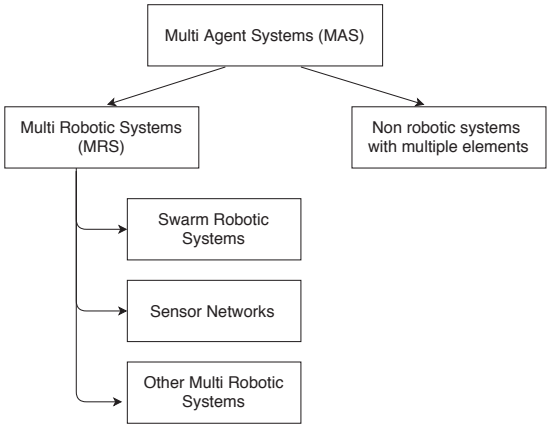


Figure 2.2: Robotic systems topology [81].

3. *Military:* Most of the militaries are working on robots to replace soldiers to save human life. There are currently war drones that fly without pilots. Therefore, the military is another useful application field for swarm robotic systems.

4. *Medical:* Curing diseases with robots is another application field. Soon, tiny robots will be able to move into our bodies to cure diseases such as cancer.

5. *Astronomy:* This application is not widely known. However, swarm robotic systems can help to analyze the universe, e.g., the black hole.

6. *Hazard zones:* Replacing humans in places that are dangerous for them is another application field for swarm robotic systems. For example, they can go into burning areas to save humans or animals.

7. *Household:* The last application field is when many tiny robots are programmed to do household chores, such as dusting, vacuuming, or washing the windows.

Swarm Robotic Systems and Other Robotic Systems Besides swarm robotic systems there are many other systems that handle multiple robots. The parent of that system is the multi-agent system (MAS). A definition of agent is given by [82]. The author says that an agent is a computation system designed to act automatically in some environment based on some rules and tasks. The agents are connected over a network to communicate with each other. For a MAS the communication is vital in this differentiation with the other robotic systems. It has to be mentioned that an agent can be either software (e.g., software bots) or hardware (e.g., robots) An overview of all multiple robot (Agent) systems is shown in Table 2.1. As we can see in this table, swarm robotic systems differ from other, primarily homogeneous (functionality is the same among robots), systems. Therefore, they are simple and cost efficient. The number of robots is much higher, and the system has a decentralized control. All the other points are similar to the other robotic systems.

Swarm Robotics Classification Because swarm robotics have different task areas, a classification of them helps developers to design and analyze them more sufficiently. The Swarm

Robotic Systems	Swarm Robotic Systems	Multi-Robotic systems	Multi-Agent Systems	Sensor Networks
Number of members	Large (as compared to other robotic systems)	Small (as compared to swarm robotic systems)	Small (as compared to swarm robotic systems)	Large (as compared to MAS and MRS)
Design and implementation of robots	Very simple. Single robots are unable to do anything significant	Single robots can perform significant parts of a task	Single robots are able to perform significant parts of a task	Nodes can be designed simple or complex
Self-organization	Yes	Yes	Yes	Yes
System Control (centralized or decentralized)	decentralized	Both	Both	Both
Homogeneity or heterogeneity	Mostly homogeneous	Mostly heterogeneous	Both	Homogeneous
Autonomy	Yes	No	No	Yes
Environment	unstructured (unknown)	structured and unstructured (known and unknown)	structured (known)	structured (known)
Movement	Yes	Yes	Mostly not	No
Robustness	yes (high)	Yes	Yes	Yes
Scalability	yes (high)	Yes (low)	Yes	Yes
Flexibility	yes (high)	Yes (low)	Yes	Yes
Cost	Low	Medium	Medium	High

Table 2.1: Differences and similarities between swarm robotic systems and other robotic systems [7].

Robotics Classification has two groups– the analysis and design methods. With the analysis method, the development happens by looking deeply into swarm robotics. That can be done by microscopic, macroscopic, sensor-based modeling based on swarm intelligence algorithm or real-robots analysis approaches. On the other hand, the design methods are based on behavior approaches, virtual and physical design, or automatic approaches. Here is a summary of the meanings of the above-mentioned terms based on the publications [83, 4, 35], and [84].

1. *Microscopic Approaches:* In the microscopic approach, the swarm robotic is analyzed based on each robot and not as a whole system. Thus, each robot is checked deeply about its functionality and features.

2. *Macroscopic Approaches:* The macroscopic approach analyzes the swarm robotic system as a whole system. Therefore, it only checks how the swarm can work together as a group of robots and not as individuals.

3. *Sensor-based Approaches:* With the sensory-based approach, analysis is done like the microscopic approach, individually. However, only collected information from the sensors or other inputs are considered here.

4. *Modelling Swarm Intelligence Algorithms:* Swarm intelligence algorithms inspired from nature can be used to model swarm robotic systems. For example, for modeling and analyzing, the particle swarm optimization algorithm is mainly used.

5. *Real-Robots Analysis Approaches:* Most of the time, simulations or theoretical aspects are used for modeling and analyzing swarm robotics. However, those approaches are significantly less familiar with real-world applications. Therefore, the last approach is the real-robots analysis approach. In this approach, real robots are used for analysis. It has the advantage of analyzing real-world approaches. However, the drawback is that it will take more time than using a simulator. Hence, each robot has to be updated when the code has been changed. As a result, it is time-consuming and not suitable for scaling.

The classification for the design methods is:

1. *Behavior based Approaches:* With this approach, the robots are kept simple in the beginning and updated or upgraded based on upcoming demands. One way of designing a swarm robotics system with the behavior-based approach is by using a probabilistic finite state machine (PFSM). PSFM is often used in swarm robotics because each time the robots receive different inputs, they go into different states. Hence the input is not predictable. Therefore, PFSM has probabilistic transition states.

2. *Virtual physics-based Design:* As the name says, the design here is virtual. A virtual environment must be set for the virtual entities to simulate the behaviors as a swarm.

3. *Automatic Approaches:* In this approach, the robots are not programmed manually. Rules are given to them, but they have to learn by testing everything independently. These have to do with machine-, deep-, reinforcing-learning in swarm robotics.

The Present State

In this subsection, we present the current projects, simulators, and real applications for swarm robotics. The current swarm robotics projects are shown in Table 2.2. In those projects, real robots are used to form a swarm. From these seven projects, the famous ones are the Kilobots from Harvard University. These are 3.3 cm tall robots that have a sensor to receive code or updates through infrared lights. Thus, with the help of infrared light, the swarm of Kilobots is scalable. Nevertheless, the price for a Kilobot is around 130 - 140€, which is expensive. Therefore, for developing a swarm robotic system, a better solution must be found that is price efficient, scalable, and development friendly.

Simulators solve the problem of cost-efficiency and fast development for swarm robotics. Cost-efficiency is handled because the robots are virtual. As a result, it is possible to simulate a massive number of robots without buying any, making the system scalable and cost-efficient. Another factor is that the time for testing algorithms on each robot is much faster because the robots are virtual, and any changes in the algorithm are copied automatically into the virtual robots. With real robots, this task needs to be done manually, one by one. Each robot must be plugged into the working station to be updated. In Table 2.3 actual simulators for swarm robotics are listed. Present real-life applications are grouped based on the swarm robotics applications introduced previously.

1. *Agriculture:* In agriculture a real application is the SAGA [55, 56, 57]. In SAGA drones monitor the agriculture field regarding weeds. It checks which places have or do not have weeds and spread them based on the demands.

Projects	Objectives	Basic Properties	References
Swarm-Bot	To form simple, reliable, flexible, scalable, self-organized and self-assembling micro-robotic systems	Robots show robustness, flexibility and are able to solve complex problems via self organization	[5, 85, 41, 86, 87]
Swarmanoid	The main goal is to design a heterogeneous distributed swarm robotic system that operates in 3-dimensional human environments	In addition to s-bots, Swarmanoid consists of hand- and eye-bots that can climb objects and fly, respectively	[88, 89, 90, 91, 92]
I-SWARM	The goal of the I-SWARM project is to build the largest robot swarm that consists of up to 1000 mini robots of size 3*3*3 mm	A single robot looks and moves like an insect. But it consists of various modules that enables it to perform significant tasks. These modules include power, electronics, locomotion and communication module	[93, 94]
SensorFly	To build an aerial mobile sensor network of robots that can perform monitoring in indoor emergency situations	The aerial robots are low-cost, autonomous, and are capable of 3D sensing, obstacle detection, path identification and adapting to network disruptions	[95, 96]
Marsbee	Exploring Mars	Consists of a colony of small flying robotic bees that can sense their environments via sensors. There is a charge station where the marsbees can recharge themselves	[97]
Kilobot	It is a low-cost swarm of small robots designed to study collective swarm behavior	Each kilobot has a programmable controller, is capable for locomotion, local communication and can sense its environment	[98, 99, 100, 101, 102]
Kobot	A circular shaped, cheap, small, and expendable robot. These features make it very suitable for various swarm robotic applications	Has IR-based short-range sensors, supports wireless and parallel robot programming and has a battery that can last up to 10 hours.	[103]

Table 2.2: An overview of current swarm robotic projects [7].

2. *Industrial:* The FIBERBOTS [62] are one example of real-life industrial applications. Here the robots work as a swarm to produce tubular forms. This production is done by wrapping themselves with a material for the form to make the tubular shape. The main action of the swarm is that the they robots do not interfere each other while they are producing. This is done with a decentralized program and without any communication; it is based on the flocking algorithm [139].

Simulator	Objective	Developers	Open source	supported languages	Supported OS	2D or 3D	Status	References
Swarm-Sim	A round based simulator developed for modeling swarm robotic systems in a 2D/3D environment.	Heinrich Heine University of Düsseldorf [9]	Yes	Python	Linux, MacOS, and Windows	2D and 3D	Active	[11, 10, 15, 14, 16, 13, 12]
Player and stage	offers free software for robots, sensors and actuators research	Brian Gerkey [104], Richard Vaughan, Andrew Howard, and Nathan Koenig	Yes	any language	Linux, Solaris, BSD and MacOSX	2D	Last update 2010	[105, 106, 107, 108]
Gazebo	offers opportunity to simulate robot swarms accurately and efficiently in various indoor and outdoor environments	Open Source Robotics Foundation (OSRF) [109]	Yes	mostly ROS (Robot Operating System)[110]	MacOS, Linux and Windows (a binary package available only for Linux)	3D	Active	[111, 112]
Robot Virtual Worlds	High-end simulation environment for students to learn programming	Robomatter Incorporated [113]	No	ROBOTC [114]	Windows and Mac	3D	Active	[115]
Teambots	Offers java classes and APIs to support research in mobile multi-agent systems	Georgia Tech's Mobile Robot Laboratory [116]	Yes	Java	Windows, NT, Solaris, SunOS, MacOS, OS X, Linux and IRIX	2D	Last update 2000	[117, 118, 119]
V-REP	A universal simulator with integrated development environment, where each item can be controlled individually	Coppelia Robotics [120]	Yes	C, Python, C++, Java, Lua, Matlab, Octave and Urbi	MacOSX, windows and linux,	3D	Active	[121, 122, 123, 124]
ARGoS	Aims to simulate heterogeneous robot swarms in real-time	Developed within the swarmanoid project [89]	Yes	ASEBA scripting language (others are under study) [125]	Linux and MacOSX	2D and 3D	Active	[126, 127]
Webots	high-quality professional mobile robot simulator used for educational purposes	Cyberbotics Ltd. [128]	Yes	C, C++, Java and from third party software (via TCP/IP)	Windows, Linux and MacOSX	3D	Active	[129, 130, 131]
Workspace	An Offline Simulation and programming platform. Offers simulation solutions for industrial and educational purposes	Watson Automation Technical Solutions Ltd. [132]	No	Many robotic languages e.g. AB G-Code and Adept V-Plus	Windows	3D	Active	[133]
OpenHRP	A virtual platform to investigate humanoid robotics	AIST [134]	Yes	C, Python, C++, Java	Linux, Windows	3D	Last update 2012	[135, 136, 137]
SCRIMMAGE	Used for the testing and comparing of mobile robotic algorithms and behaviors	Georgia Institute of Technology	Yes	Python, C++	Linux, MacOS	3D	Active	[138]

Table 2.3: Comparison and differences between several swarm robotic simulators [7].

3. *Medical:* For the medical application, there are not any perfectly capable applications, hence, the robots should be nanosized to cure diseases inside the body. However, the Max-Planck institute developed programmable millirobots [65, 66], that can swim, walk, transport, or jump. If in the near future those robots can shrink to nanometer size, they will be helpful in to curing diseases inside the human body.

4. *Hazardous Zones:* The snake robot [67] from the Carnegie Mellon University in Pittsburgh can save lives by finding survivors moving through narrow spaces after a disaster. For example, in 2017, this snake robot was used in Mexico after an earthquake to search for humans among the ruins. It is equipped with sensors and a camera to analyze and capture the situation. Therefore, it is instrumental in hazard zones.

5. *Military:* There are a few swarm robotics real-life applications for the military [71, 72, 73]. Using robots instead of human armies have the following advantages. First, having robots in a real war will save soldiers' lives. Additionally, robots are more precise and efficient in targeting aims than humans. Further, they are bullet-proof and resistant against chemical gases. To sum up, having robots instead of human soldiers has notable advantages.

6. *Household:* Vacuuming robots have already become fairly commonplace. However, there is another real-life application termed "MAB" [76]. In this project, small robots fly as swarms to collect dirt and dust and disposing them into a trashcan. It is an exciting project for households.

7. *Astronomy:* Using swarm robotics systems for astronomy is an emerged real-life application field. The eStart [52] is one example for real-life applications. Estate is a swarm of heterogeneous robot telescopes. They all work together for a common result, while monitoring space. There are many more astronomy swarm robotics fields that are described in [54].

The presence of swarm robotics shows a lot of exciting projects, simulators, and real-life application fields. Nevertheless, the present is the new future, so let us see what the future of swarm-robotics holds.

The Future of Swarm Robotics

For swarm robotics, there are many visions of innovations for the future, which is presented here. The first one is nanobots, i.e. very tiny robots in the range of nanometers. They are nearly the size of a virus, with a diameter between 20-400 nm. These nanobots can be injected into the body to find diseases undetectable for humans. For example, they may be on a mission to find a cancer cell. When they see it, they will coat and destroy it. However, the big challenge here is building robots that are nanosized. Considering, that currently, the size of a transistor varies between 20-40 nm, building nanobots might be possible soon.

Another example of the futuristic application of swarm robotics is armies of robots. Here the robots move together as a swarm to defeat enemies. The challenge here is to build these robots to be capable of walking as a swarm and creating the software to coordinate this movement,

such as flocking without any centralized unit of control. Instead, using those robots for military purposes can replace human laborers, e.g., for parcel delivery or transportations.

There are still more challenges and future applications for swarm robotics. Such as using the snake robots for finding survivors and in medicine, industry, or manufacturing. Another example is swarm robotics in space [140]. The swarm is in space to conquer and research everything that is near the earth. The same is possible for inspections in areas here on earth (drainage channels).

The swarm robotics research area can change to a more innovative place where robots are produced and programmed to be used for challenges for humankind, either risky, dangerous, or inefficient. Hence, hardware technologies, such as sensors or CPUs, are becoming cheaper and smaller. Therefore, swarm robotics is still an emerging field of research that needs to be turned into real-life innovations. Additionally, for the future of swarm robotics, some challenges must be kept in mind. These challenges regard software and hardware in swarm robotics. Both software and hardware should be designed to be flexible, robust, and scalable. That means that the software should handle scales of robots, environments, and losses within the swarms. However, there must be enough gadgets in the hardware to address changes in the environment (e.g., obstacles) or to take away broken robots in a swarm or sensors to recognize scalability within the swarm. To sum up, we need, in the future, enough gadgets and sensors to provide input for the software to handle the three significant rules of swarm robotics, scalability, robustness, flexibility.

In this paper, we introduced the past of swarm robotics, explained how swarm robotics was inspired by nature, and who the pioneers were in this research field. Then, we talked about the presence of swarm robotics. We presented the actual projects, simulators, and real-life applications. Lastly, the future of swarm robotics was imagined; we proposed our vision for swarm robotics innovations, and considered which software and hardware issues need to be solved. As a result, swarm robotics is an emerging scientific area that can become more innovative with new technologies, such as 5G Internet and Artificial Intelligence (AI).

2.1.2 Importance and Impact on Dissertation

This survey covers the first module M1 and the contribution C1.1 for this dissertation, which is to do a comprehensive literature survey about robot swarms. This survey summarizes 217 publications and provides knowledge of robot swarms. Therefore, we could answer RQ1.1 and solve the first problem statement i.e. gaining knowledge about robot swarms. This survey is essential because it gives all the necessary information about robot swarms, and thus, builds the foundation of knowledge for this dissertation.

2.1.3 Contribution

There are many papers and surveys about swarm robotics, and most of them give an overall perspective of this topic. Nevertheless, none of them have the focus on presenting swarm robotics starting from its roots (the past), moving slowly to its branches (present), and finally

taking up fruits (the future). Therefore, this survey has a comprehensive overview of swarm robotics' past, present, and future, which makes an outstanding contribution to the scientific world.

2.1.4 Personal Contribution

The contribution of Ahmad Reza Cheraghi is the motivation, structure, and methodology of this study. In addition, he finalized the writing of this paper. Under his supervision, Sahdia Shahzad researched the literature, prepared the summaries, and wrote the first draft of this paper. Kalman Graffi was continuously involved in discussing the scientific approach and provided critical revision of the paper.

Chapter 3

Developing a State-Of-The-Art Simulator for Robot Swarms

The theory of robot swarms has been discussed. It is now necessary to explain the practical work. This chapter describes how we developed the simulator Swarm-Sim. We first start with the survey of simulation tools for peer-to-peer, ad-hoc, and opportunistic networks. There has been plenty of work done on simulators [141, 142, 143, 144]. We examined Twenty-five network simulators to find out their current state, usability, and fame in the science world. This survey gives brief information about these network simulators, and it helps us find out if there is any network simulator, we can use for a swarm simulator. Nevertheless, Swarm-Sim was inspired by a simulator made by the University of Paderborn [17, 18]. It is written in Python, which is easy to learn and contains a lot of useful libraries. Their simulator uses a triangular grid with only two items: tiles and particles. However, we took the idea of the Paderborn simulator and developed a new simulator. Our simulator, Swarm-Sim, provides an interface to develop a new algorithm for robot swarms and add additional capabilities for tiles or particles. Additionally, the simulator can be configured easily, and it is 2D and 3D. Therefore, we came up with the idea to write a new simulator based on Python.

3.1 The State of Simulation Tools for P2P Networks on Mobile Ad-Hoc and Opportunistic Networks

This section summarizes the contributions and gives a verbatim copy of our paper [8].

Ahmad Reza Cheraghi, Tobias Amft, Salem Sati, Philipp Hagemeister, and Kalman Graffi
"The State of Simulation Tools for P2P Networks on Mobile Ad-Hoc and Opportunistic Networks"
In: *Proceedings of the International Conference on Computer Communication and Networks(ICCCN). 2016.*

3.1.1 Paper Summary

We use simulators to predict occurrences that are difficult to predict in real-life, or are time and cost-consuming. In the world of data transfer, the Internet is a big player. This network connects many computers, typically through a client-server architecture. That means that each computer is the client which connects to a server to receive data. This type of service is called a centralized network; everything is managed and handled centrally. A disadvantage of a centralized architecture is that it is vulnerable. If a server is shut down, collecting information is no longer possible. Therefore, another approach must be found. The most common approach is the peer-to-peer (P2P) approach, the decentralized way.

P2P is a network built on an existing network, such as the Internet [145, 146, 147, 148, 149]. Here each computer becomes a peer, and they are all connected directly with each other without using any server. The connection within a P2P network works when each peer knows the IP addresses and the specified port number of another peer. However, these peers are connected to other peers through the Internet without using the standard server-client mechanism.

However, access to centralized services from the Internet is at risk because they have vast connection points such as routers and servers. Additionally, each country has its connection point (Gateway), which leads to the rest of the world. Therefore, a government can decide to encapsulate itself from the rest of the world, which has been witnessed many years ago in Egypt and a few years ago in Iran. Therefore, a new way of the network is vital.

The mobile ad-hoc network (MANET) is one way to overcome the vulnerability of the Internet [150, 151, 152]. The MANET is built on a mobile device. Mobile devices can be, for example, our smartphones. When smartphones are connected through Wi-Fi or Bluetooth, they create a MANET. Other smartphones can connect themselves ad-hoc to the MANET, too. Uniquely defined routing protocols for MANET manage the data transfer between the smartphones. This means that each device acts as a router as well. However, there must be a connection between the sender and the receiver in a MANET. Otherwise, MANET routing protocols drop the messages after a while. The opportunistic network (OppNet) can overcome this obstacle. The OppNet belongs to delay-tolerant networks. An OppNet is some MANET. However, it is slightly different. The routing protocols are defined to tolerate delays without dropping the messages, and a connection between the sender and receiver must be provided. Messages sent

within the OppNet are passed between the devices, hoping they will reach their destination sometime.

The combination of the P2P, MANET, and OppNet makes the network less vulnerable, where censorship is no longer possible. MANET and OppNet provide an underlay that is created and handles by devices (peers). Because, with devices, everything is decentralized, there is no given infrastructure, and everything occurs ad-hoc. Further, P2P is the overlay that manages all the querying, fetching, sharing, and searching of messaging. As a result, to have a nonvulnerable communication method, a combination of these three underlays is one solution.

However, it is necessary for simulators to test and evaluate extreme situations within the networks, such as data flooding or bringing changes in the size of the peers. Therefore, it is essential to understand which simulators are on the market and can simultaneously support P2P, MANET, and OppNet.

Therefore, this paper aims to give an overview of the simulator, which can simulate P2P, MANET, or OppNet. We analyze them based on the number of citations from scholar.google.com if they are active or inactive, open-source or close, or event or round based. Finally, it was essential for us to know which could support all three layers. Moreover, the conclusion is that NS3 and PeerfactSim.Kom is the preferred simulator to help all P2P, MANET, and OppNet. Because they support mobility, and the most necessary protocols exist or can be developed.

Nevertheless, all the simulators became too complicated because of their outdated programming and languages which are inefficient for programming today, such as C, C++, or Java. Therefore, it is necessary to think about creating a new simulator, which is easy to learn and can write and prepare algorithms for testing, especially in robot swarms.

3.1.2 Importance and Impact on Dissertation

One basis to gain knowledge is research. This survey is vital for module M2 because it provides information about the actual state of the simulation tool before starting to develop a new simulator, which is the contribution C2.1. With this survey we answered RQ2.2 by presenting the actual state of 25 simulation tools for peer-to-peer, mobile ad-hoc, and opportunistic networks. It provided the necessary information about those simulators and impacted the decision to develop a new simulator.

3.1.3 Contribution

Simulators are essential tools to evaluate the new and emerging algorithms before they go to production. Therefore, it is necessary to know about the actual state of those evaluation tools. This paper presents an overview of simulators for P2P, MANET, and OppNet. The simulators are chosen based on three criteria. First, the number of citations on scholar.google.com; this is analyzed to gain knowledge on how often the simulators are used in research papers. Second, if the simulators are active, i.e., how long has it been since the latest version has come out and does it have a vibrant community? Finally, it is important to find out which of these

simulators are helpful to simulate a combination of these three networks. Therefore, the main contribution of this paper is the overview and comparison of P2P, MANET, and OppNet and to find out which simulators are helpful for the combination of these three networks, which has not yet been explored in science. It is an outstanding contribution to the science world.

3.1.4 Personal Contribution

Ahmad Cheraghi analyzed the existing work on MANET simulators, conducted the paper submission, and organized the final version of this paper. Tobias Amft, the author of this thesis, collected and summarized information about existing peer-to-peer simulators. Furthermore, he organized and shaped the first version of the paper. Salem Sati reviewed OppNet simulators and summarized them. Philipp Hagemeister motivated the application case of the paper. Kalman Graffi contributed to this paper's motivation and methodology and guided the production.

3.2 Swarm-Sim: A 2D & 3D Simulation Core for Swarm Agents

This section summarizes the contributions and gives a verbatim copy of our paper [9].

Ahmad Reza Cheraghi, Karol Actun, Sahdia Shahzad, Kalman Graffi
"Swarm-Sim: A 2D & 3D Simulation Core for Swarm Agents"
In: *Proceedings of IEEE IRCE 2020: The 3rd International Conference of Intelligent Robotic and Control Engineering.* 2020.

3.2.1 Paper Summary

Simulator tools help to test and evaluate algorithms. This paper presents the round-based 2D/3D simulator "Swarm-Sim." Swarm-Sim provides an evaluation environment with an interface to quickly build scenarios and solutions for use-cases for robot swarms. As we already answered the RQ2.1, we know that using real robots as a swarm is expensive and time-costly. Therefore, we built Swarm-Sim.

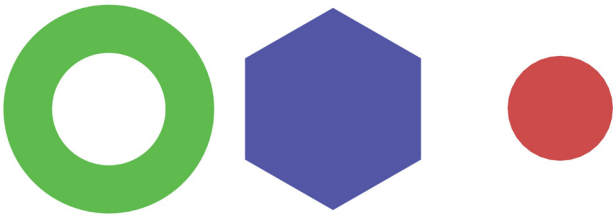
Nevertheless, we did advanced research about actual swarm simulators on the market. We found 11 simulators. However, most of them aim to simulate realistic robots with advanced and complex graphics. Additionally, most of them bring their main attributes in a continuous space, which will create minor errors and inconsistencies, perhaps leading to floating-point rounding errors. As a result, Swarm-Sim uses a quantized and modular space. Its performant and straightforward visualization makes real-time 3D simulations of large amounts of robots possible. Moreover, having Python as its primary programming language makes learning easy, and it is Operating Systems independent.

The idea of Swarm-Sim originally comes from the University of Paderborn [17, 18]. They made a simulator for simulating particles that split into two like a liquid when they move from one position to another. This kind of movement the researcher of Paderborn termed the Amoebot. With this Amoebot model, the possibility of making a swarm is not given. We generally build a new simulation system that is simple and efficient.

Swarm-Sim is simple because it provides elements, termed matters (Fig. 3.1), with primary attributes making a simulation of swarms very easy, taking away the complexity and saving time. These matters are either passive or active. Passives are locations or items, and the active ones are agents.

Locations (Fig. 3.1a) are used to mark coordination points on the playing ground of Swarm-Sim to share information. Compared to the other matters, location has only one attribute: memory to store data. Additionally, it cannot be taken or dropped by agents. A real-world example can be a written text with chalk on the ground to transfer information.

Another matter in Swarm-Sim are Items (Fig. 3.1b). Items can be taken or dropped by an agent. Examples of an item can be bricks to make obstacles or a playing ground, such as an



(a) Location (b) Item (c) Agent

Figure 3.1: Visualization models of matters for a triangular grid [9]

island on which agents can move on.

However, the last and essential matter is the agent (Fig. 3.1c). An agent is the active element in Swarm-Sim and has the most attributes. It can create and delete items and mark locations to read and write pieces of information. Additionally, it can take and drop items. An essential feature is that it can move between each location point in Swarm-Sim. It can also read from and write on matters, including communication with other agents. Finally, it can scan its environment for other matters. Agents are the most crucial matter in Swarm-Sim because they are active and represent the swarm's entities.

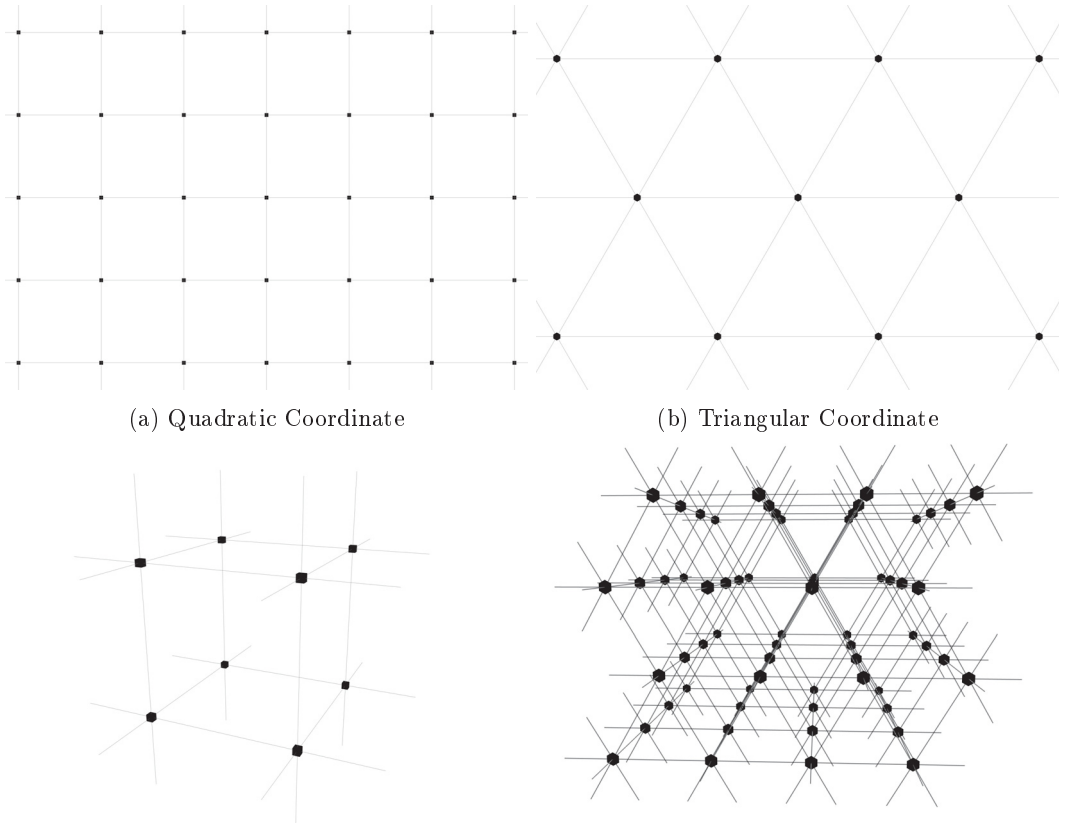
Besides the matters, Swarm-Sim provides a playing ground termed grids. Grids consist of coordinates of the so-called locations or location points. These locations are connected to neighbor locations and thus builds up the grid. The distance between two location is termed hop. Matters can be positioned on the locations, and agents can move within the grid by moving from one location to another adjacent location. In Swarm-Sim, the grids are either in 2D or 3D, and four basic grids (Fig. 3.2) with different amounts of adjacent neighbors are provided.

The four basic Swarm-Sim grids are Quadratic, Triangular, Cubic, and Cubic Close-Packaging. The quadratic coordinate grid is built out of squares, and each location point has four adjacent points, whereas the triangular Coordinate Grid is formed out of triangles, and each location has six adjacent locations points. Both grids are for 2D simulations. However, by extending those by the z-axes, the grids change to 3D. Quadratic becomes the Cubic grid with six neighbors, and the Triangular Grid changes to the Cubic Close Packaging grid with eight neighbors.

Next, it is crucial to discuss the architecture and interface of Swarm-Sim. The architecture in Fig. 3.3 shows the overall look of the Swarm-Sim. It consists of five modules: the Core, Configuration File, Evaluation Data, Scenario, Solutions, and Interactive Window.

The Swarm-Sim Core holds all the essential parts of the simulator that should not be touched. These are the matters, locations, items, and agents. Additionally, it contains the visualization engine, config-parser, and most essentially, the World-Interface. The World-Interface is vital because it is used to write solutions and scenarios to simulate swarms in the Swarm-Sim.

The World-Interface provides methods for getting information about the actual state of or for manipulating the simulation. It provides information, e.g., about the actual round number, the number of created matters, and their positions. Generally, it is a database that keeps all the



(a) Quadratic Coordinate (b) Triangular Coordinate

(c) Cubic Coordinate (d) Cubic Close-Packing Grid

Figure 3.2: Three-dimensional grids [9]

data regarding the actual simulation status. It also provides methods to add or delete matters before or during the simulations. Additionally, the World-Interface contains information about the simulator's actual state. It tells the actual simulator round. With this information, we can write a solution to explain in what round something should happen. Next, the World-Interface allows the termination of the simulation. It is essential to know when a solution reaches a prosperous state in order to terminate the simulation. Thus, the World-Interface provides an interface to stop the simulation. Further, the information of the grid is provided by the World-Interface, too. As discussed, Swarm-Sim provides different grids. Thus, this interface provides the type of grid and its coordinate system.

However, one main task of the World-Interface is to create scenarios and solutions. The Swarm-Sim works with two Python files which must be defined first. The first one is the scenario, in which the initial state of the simulator is defined. Furthermore, the second one is the solution, in which algorithms for simulating swarms can be written.

Listing 3.1: Example of a simple scenario

```
def scenario(world):
world.add_agent((0.0,0.0,0.0))
world.add_item((2.0,0.0,0.0))
world.add_location((4.0,0.0,0.0))
```

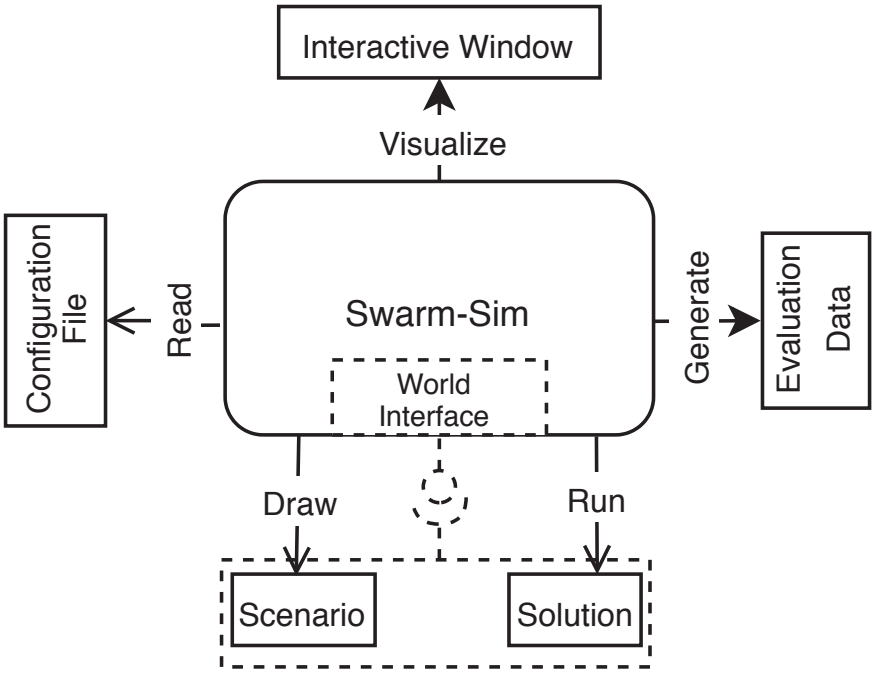


Figure 3.3: Swarm-Sim Architecture [9]

A scenario is the starting face of the simulation. A scenario file is a Python file where we write our starting state of Swarm-Sim. In this file, we can draw how the matters should be positioned. We use the World-Interface to position matters in the scenario file. In Algorithm 3.1 we show a simple example of a scenario. In this scenario, we added an agent, item, and location on different coordinates, by using the methods add_item(coordinates), add_agent(coordinates), or add_location(coordinates). The visualized output of the scenario is shown in Fig. 3.4. Nevertheless, our aim is for agents within the swarm should to do actions to solve problems. Thus, it is necessary to define solutions.

Listing 3.2: Example of a simple solution

```
def solution(world):
for agent in world.agents:
items = agent.scan_for_item_in(hop=1)
if len(items) > 0:
agent.take_item_with(items[0].get_id())
direction=random.choice(world.grid.get_directions_list())
agent.move_to(direction)
direction=random.choice(world.grid.get_directions_list())
agent.drop_item_in(direction)
```

For simulating a swarm, it is necessary to define a solution. In the solution, we can write an algorithm to explain what the agents should do and outline their limitations. The World-Interface provides access to all the matters that we already defined in a scenario. Thus, it helps us to write an algorithm for swarms in the solution. A simple solution example is shown in Algorithm 3.2. In this solution, all the agents scan the adjacent locations to find an item. If the scanning is successful, the agent takes the founded item, moves randomly in any direction,

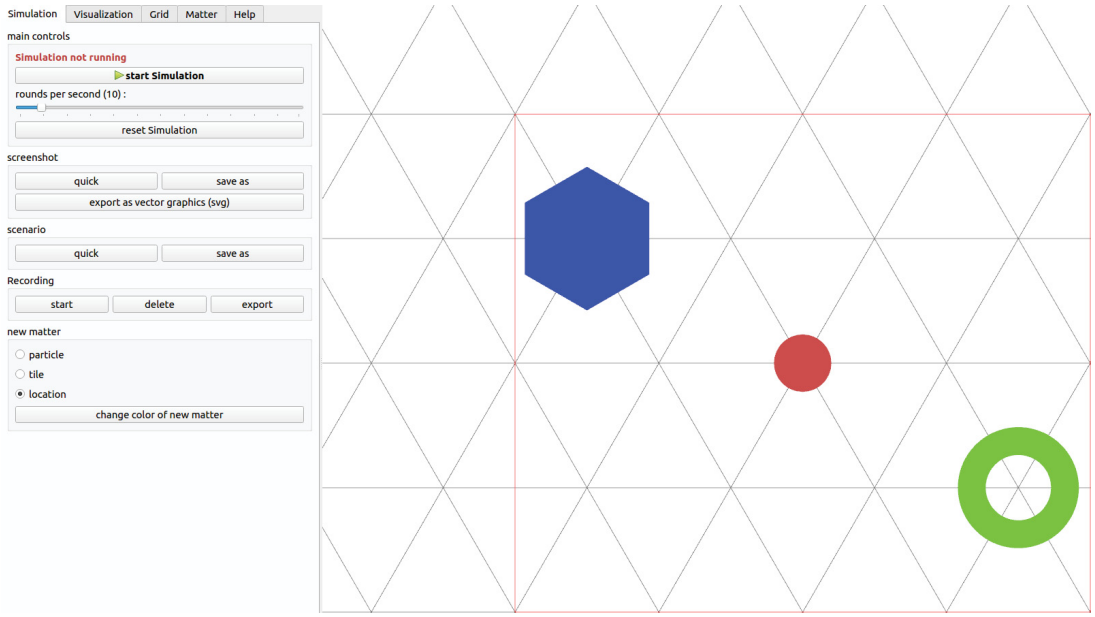


Figure 3.4: Example of a 2D Scenario [9]

and drops it on a randomly chosen adjacent location. As we mentioned earlier, Swarm-Sim is a round-based simulator, which means that the solution is called repeatedly after each round until the maximum number of rounds is reached. The maximum round number, the developed scenario-, and solution-file are set in the config file. Thus, before running Swarm-Sim, it is necessary to set up the simulator with the help of the config file.

The config file in Swarm-Sim allows one to set up the simulator before running. We can set different parameters, for example, the maximal round number before the simulation is terminated; we can decide if it should simulate in 2D or 3D by defining the necessary grid type, or if the grid should be with or without borders. Most importantly, we can choose which scenario and solution should be used. To sum up, with the help of the config file, many setup possibilities are given, making the Swarm-Sim flexible and easily configurable. However, an advanced simulator should have outputs for showing results to evaluate different solutions.

After each terminated or ended simulation, the Swarm-Sim generates evaluation data. The evaluation data are three comma-separated value (CSV) files. First, for each round, second for each agent, and last an aggregation of round and agent files. The round file stores all the global data of the simulator. For example, the numbers of matters available, the total moves made by the agents, the take and drops of matters by the agents, and many more. Next is the CSV file for agents. The agent evaluation data shows the result from the view of an agent. It provides the actions of each agent. For example, it shows the movement steps or the total number of drops or takes. The last CSV file is the aggregation one. Here, all the agent and round CSV file data are aggregated to summarize the whole simulation. The summary includes, e.g., the success rate, the average, maximum, and minimum movement steps, and many more. In summary, the evaluation data is necessary for showing the result as a CSV, to be aware of how the programmed solution could perform. However, it is necessary to visualize those results,

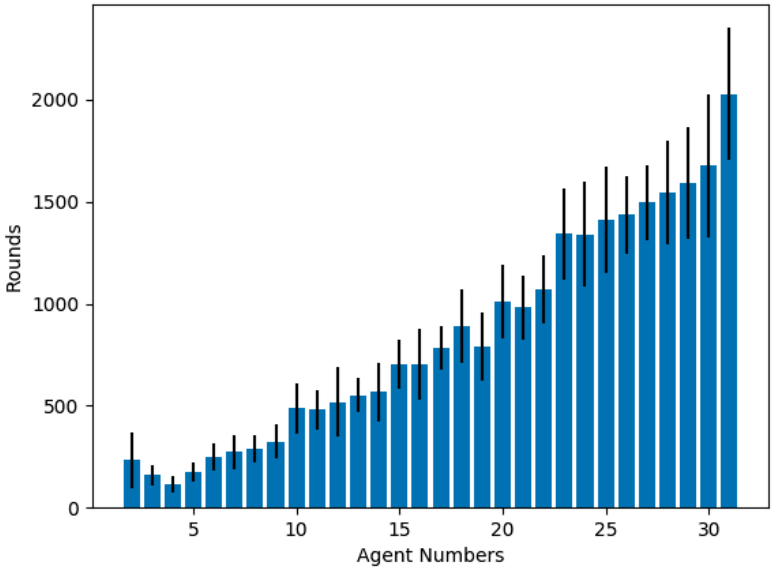


Figure 3.5: A Swarm-Sim plot [9]

which is also possible with the Swarm-Sim.

Swarm-Sim also provides the possibility to generate plots such as bar charts or diagrams from the CSV-Files. These visualized results are generated either in a PNG- or PDF format, used for presentation or publication. An example of a diagram is shown in Fig 3.5. Here we can see the rounds on the x-axis and the total number of agents on the y-axis. In this example, we measured the average time (in rounds) different sizes of self-charged phototactic (moving whenever sensing light after got charged by the light) robot swarms needed to reach a given finished line. As we can see, the more robots in the swarm, the longer it takes for the robot swarm to reach the finish line. Nevertheless, plotting a CSV file helps to visualize what was happening during the simulation. Looking at the results after the simulation might be interesting, but what happens while the simulation is running is even more interesting. For this purpose, Swarm-Sim provides a graphical user interface.

The Swarm-Sim GUI, as shown Fig. 3.6 provides two windows. The right part of the GUI is to watch the simulation while running. In the beginning, it shows the starting state of the simulator based on the scenario. When the simulator starts, it shows the states that are caused by running the solution. The left part of the GUI provides an ad-hoc configuration of the Swarm-Sim. Here it is possible to change some parameters, add additional or delete matters, change their colors, and many more. It is even possible to change the visual size of the grid's line and points. It also can capture videos or take a screenshot of the simulation animation. Finally, it is possible to start and pause the simulation and change the speed of the simulation and animation. These provided options by the Swarm-Sim GUI are beneficial in changing some configuration parameters while running the simulator.

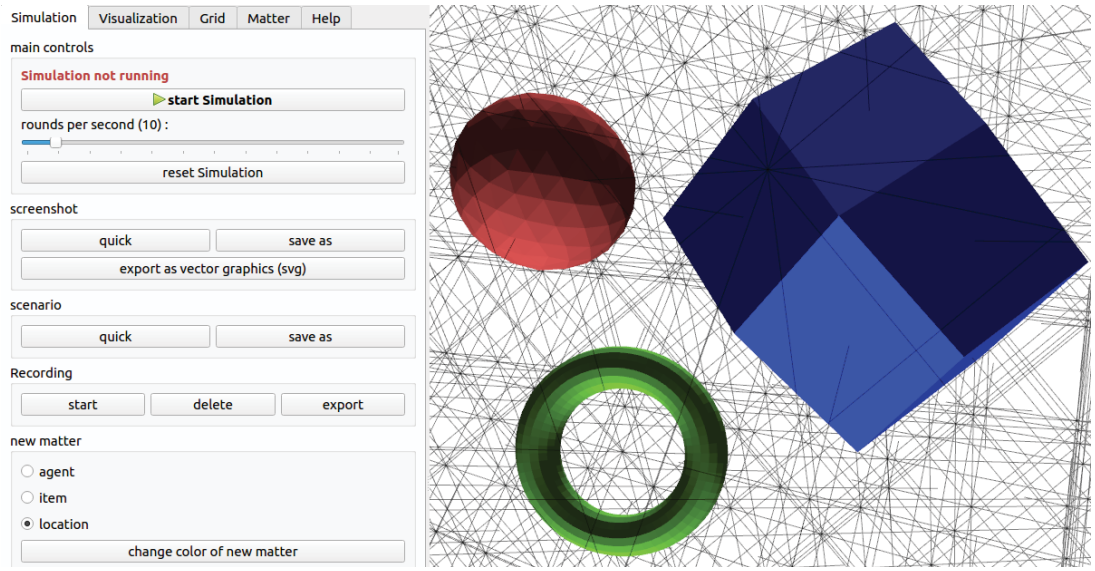


Figure 3.6: Swarm-Sim GUI [9]

Swarm-Sim provides a unique simulation environment with simple, adjusted, and customized matters. The grids provide the agents with a strict pattern movement because of their fixed locations and equal distances. Creating scenarios and solutions becomes very simple by having the World-Interface, particularly for getting information about the current state while the simulation is running. Additionally, Swarm-Sim generates evaluation data in the form of a CSV file or plot. Moreover, the GUI of Swarm-Sim allows for visualizing the running simulation either in 2D or 3D. The config file makes configuring the simulation simple without changing anything in the code. Lastly, Swarm-Sim is open source and built with Python. Thus, it enables building a community to make Swarm-Sim much more remarkable and efficient, and with Python as a programming language, it is uncomplicated to develop and evaluate new swarm algorithms. The source code of Swarm-Sim can be downloaded from the website `http://swarm-sim.com`.

With Swarm-Sim, we can implement and simulate many use-cases, especially for robot swarms. However, we implemented, simulated, and evaluated exited use-cases for this dissertation. In the following chapters, we present them more precisely.

3.2.2 Importance and Impact on Dissertation

The simulator Swarm-Sim builds the foundation of this dissertation and the basis for the following algorithms. This paper is the central part of module M2. It impacts the dissertation because, with Swarm-Sim, it contributes C2.2, i.e. the basis to develop, test, and evaluate algorithms from which we published seven papers. We have already answered RQ2.1 in chapter 1.2 to use simulated robots. To answer RQ2.3, in this paper, we have modeled the natural swarm mathematically. Additionally, we present the way of developing a robot swarm simulator, which is the answer to RQ2.4. This paper has a significant impact on this dissertation

because it provides all the information about the simulator Swarm-Sim, which is necessary to build and simulate nature-inspired algorithms.

3.2.3 Contribution

There are a few swarm simulators on the market. They are written in C, Python, Java, and many other languages. They are complicated to learn, and only one of them is open-source. In contrast, our Swarm-Sim is open-source, it is easy to learn because it is written in Python, and has three matters with minimum required properties. It is expandable because it provides a framework for adding new matters or grids. Additionally, it is flexible in simulating different scenarios such as the foraging of ants, flocking of birds, phototactic behavior, and many more. Therefore, it provides a diverse playground for scientists worldwide to test and evaluate different robot swarm algorithms. Nevertheless, Swarm-Sim's core inspiration comes from the work of Prof. Dr. Christian Scheideler and Dr. Robert Gmyr at the University of Paderborn [17, 18].

Their simulator uses a triangular grid and two kinds of matters: Tiles and Particles. Tiles are the same as items, and particles are the same as agents. Additionally, their simulation uses the Amoebot model for their particles. This model uses semi-liquid particles, and therefore they have an additional state by moving from one location to another. This state allows a particle to be between two locations before it is in its completed positioned on the next location. Generally, it means that a particle needs an additional round to move from one location to another. Additionally, the Paderborn simulator does not have a GUI that allows configuration and changes in the scenario. Further, it does not support 3D visualization. Our Swarm-Sim simulator differs from the one developed at the University of Paderborn in that it is flexible, expandable, and supports a wide range of use cases, scenarios, and swarm algorithms. It does not use the Amoebot model, therefore, it is faster.

Further, Swarm-Sim provides, in addition to the triangular grid, three more grids in 2D and 3D. It provides versatile capabilities of locations, items, and agents, and the ability to cover all considered use cases. It has an ideal workflow from easy implementation, configuration, and evaluation resulting in CSV evaluation results and plots. Finally, it is open-source and free, thus adding value to the open-source community and science world. As a result, Swarm-Sim is a detailed simulator for simulating robot swarms, and thus, it is beneficial to the science community.

3.2.4 Personal Contribution

Inspired by the University of Paderborn, Ahmad Reza Cheraghi designed and developed the simulator Swarm-Sim. He also drafted and wrote the most significant parts of this work. Under his supervision, Karol Actun renewed the Swarm-Sim GUI. He added new features, such as changing the size or colors of matters or taking screenshots of the simulation animation. Finally, he extended Swarm-Sim to run in 3D. He contributed the introduction and some parts of the related work for this work. Next, Sahdia Shahzad's contribution is the comprehensive comparison of the simulators and some parts of the related work. Finally, Kalman Graffi supervised the overall development of the Swarm simulator, added several design decisions, and provided a critical revision of the paper.

Chapter 4

Coating Objects with a Swarm of Robots

Now that we know about the simulator Swarm-Sim and how it works, we come to the robot swarm algorithms. This chapter presents two algorithms for coating objects with robot swarms. The first is the leader coating algorithm. In this algorithm, a robot within the swarm becomes the leader. It first scans the object to know where to coat the robots and how many are necessary. After that, it takes each robot to cover the object. It is straightforward to coat simple robots. However, it becomes complicated with cave-shaped objects and as the number of robots increases within the swarm. In the second section, we show the general coating algorithm. The general coating algorithm uses the whole robot swarm to coat the object, and it can coat an object of any shape.

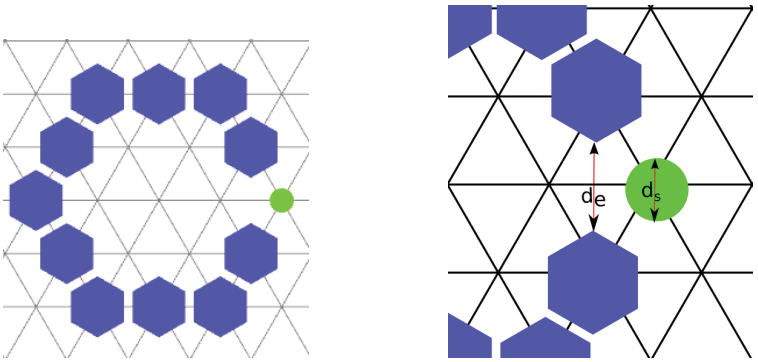
4.1 A Leader Based Coating Algorithm for Simple and Cave Shaped Objects with Robot Swarms

This section summarizes the contributions and gives a verbatim copy of our paper [10].

Ahmad Reza Cheraghi, Kalman Graffi
"A Leader Based Coating Algorithm for Simple and Cave Shaped Objects with Robot Swarms"
In: *Proceedings of IEEE 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS))*. 2020.

4.1.1 Paper Summary

This paper proposes an algorithm for object coating with a leader within a swarm of robots. The leader has, compared to the other robots, more computational power. It can recognize an object, move towards it, scan its surroundings, and plan how the swarm members' coating should be. Additionally, the leader can take and drop robots.



(a) Cave shaped object

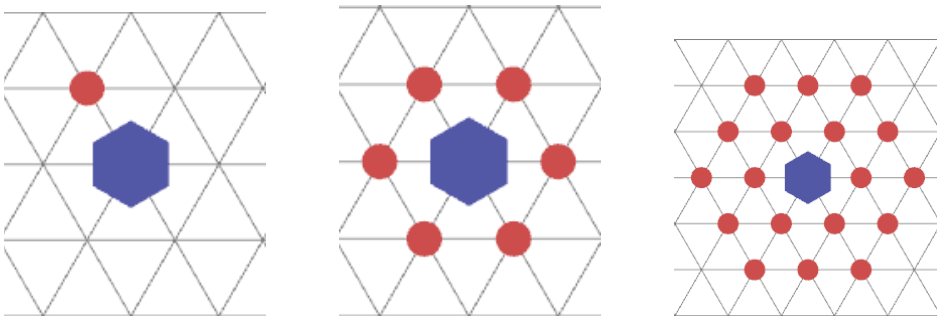
(b) Trail entrance- (d_e) and subjects-diameter (d_s)

Figure 4.1: A Cave-shaped Object Formation and its Diameter [10].

This paper proposes an algorithm for cave-shaped object coating with one leader within a swarm of robots. The ability of the swarm is for all robots to move together in any decided direction. However, the leader has more capability than the others. It can recognize an object, move towards it, scan its surroundings, and plan how the swarm members' coating should be. A cave-shaped object is shown in Fig. 4.1a. The structure of a cave-shaped object is that it includes an entrance, that leads inside the cave, which must be coated, too. Therefore, the leader needs to recognize this entrance, go inside, and scan it. An entrance of a cave-shaped object has the following condition: $d_s <= d_e < 2d_s$, which means that the entrance diameter d_e must be bigger or equal to the leader's diameter d_s and smaller than twice of the leaders' diameter i.e. $2d_s$. This information is necessary for the leader algorithm to recognize an entrance (Fig. 4.1b). However, coating a cave-shaped object brings some challenges, which will be elaborated on later, step by step. Before this, we will define what a suitable coating is and how it can be tested. The leader aims to coat an object with all the robots within the swarm. The robots must be as close as possible to the object, minimizing their distance from the object. Therefore, the distance of all the robots within the swarm must be smaller or equal to all free location distances. A coating algorithm will thrive when the following condition is fulfilled:

$$max(DOS_O(sl)_{\forall sl \in SL}) \leq min(DOS_O(fl)_{\forall fl \in FL}) \tag{4.1}$$

The equation has the function max and min, which selects a set's maximum or minimum distance. Next, it uses the function DOS_O, which calculates the distance of either a robot sl or a free location fl towards the objects. Thus, the equation checks if the maximum distance within the swarm SL is smaller or equal to the minimum distance within all free locations FL. If this condition is true, the coating is valid. Depending on the number of robots, we define three types of coating validations as shown in Fig. 4.2. The red dots are the robots, and the blue hexagon tiles build the object. The first valid state is "Legal," which means that the object is not entirely coated because there are not enough robots. The "Ideal" state is the second one. This time the number of robots is sufficient to cover the whole object. The last valid state is the "Layered" state. Here, the number of robots exceeds the number of coating positions. Thus, robots start to coat on those robots that have already coated the object. Depending on the number of robots, the layered coating can continue to build more layers of robots. Now that we know when a coating is valid, we must now consider the algorithm.



(a) Legal (b) Ideal (c) Layered

Figure 4.2: Valid Coatings [10].

Coating an object is simple when the object is as simple as it is shown in Fig. 4.2. It is a one tile object and has all it needs to coat it. The leader sees the object, scans all the black location coordinates, then takes each robot from the swarm and positions it on the scanned location. It scans that layer and starts the coating again. Whenever a layer is completely coated and robots are still left for coating, this procedure repeats again and again, until no robots are left. However, for a cave-shaped object, this coating procedure must be extended. These extensions occurred while evaluating the leader-based coating. Therefore, we first describe the evaluation setup and how we extended this algorithm after each evaluation to achieve a 100 percent success rate. Our evaluation environment is straightforward. We use the simulator Swarm-Sim, and a scenario: a cave-shaped object with a robot swarm. In the first evaluation, we start with one leader to see if it can scan the object and coat itself. After that, we increase the swarm by one robot and restart the simulation. We do this evaluation on up to 114 robots. There is no time limitation, and the coating algorithm stops when the leader does the coating and coats itself. The metric that we use in this simulation is the success rate. We want to know that after the coating is finished, the coating fulfills the criteria of the Equ. 4.1. Therefore, the simulator checks before termination if the coating is valid based on the Equ. 4.1. With a valid answer, the evaluation is marked successful.

To describe the coating algorithm, we start to coat a simple object. Later, we test this algorithm with a cave-shaped object. We define a simple object as an object that has no entrance or complex structure (something that might lead the robots to go inside). Coating such an object is simple, too. The flowchart for coating with a leader can be seen in Fig. 4.3. The flow chart has five stages. In the first stage, the leader moves towards the object. When it reaches it, it scans the object's surroundings until it comes to an already observed location, termed "repeated location." Then the leader comes to the stage of coating. Here the leader takes and drops the other robots. After the leader finishes coating the scanned locations, it checks if any robots are left for coating. If so, it goes to the "scan outside" stage. In this stage, the leader scans the surroundings of the coated robots and then starts coating again. These two stages, "Scan outside" and "Coating," are repeated until there are no coating robots left; then the leader coats itself. After that, the simulator checks if the coating condition is granted, and then it ends. We tested this simple coating algorithm on a simple object with 100 robots, and the result was a 100 percent success rate. Now that we know how the algorithm for coating simple objects works on all simple objects, let us see if it runs on a cave-shaped object.

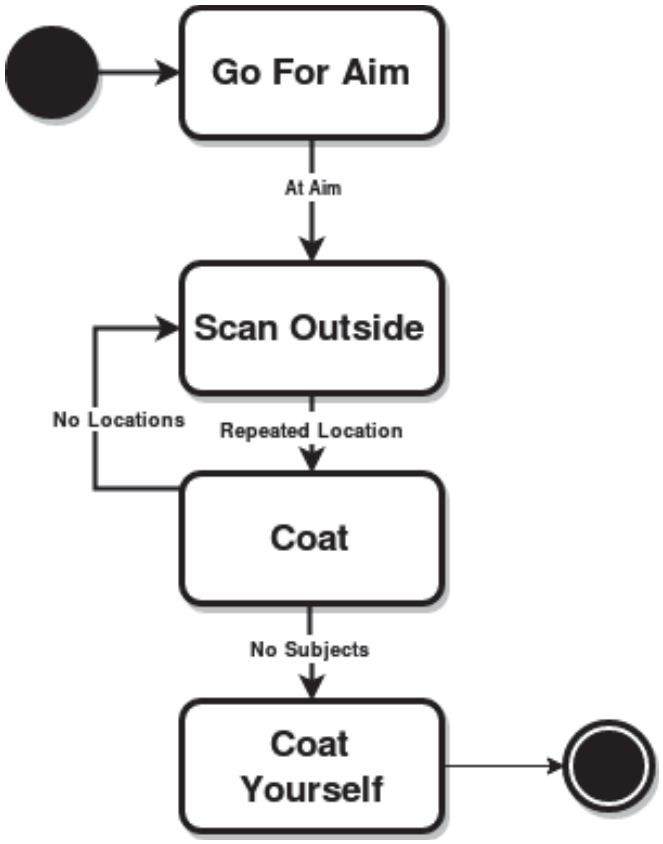
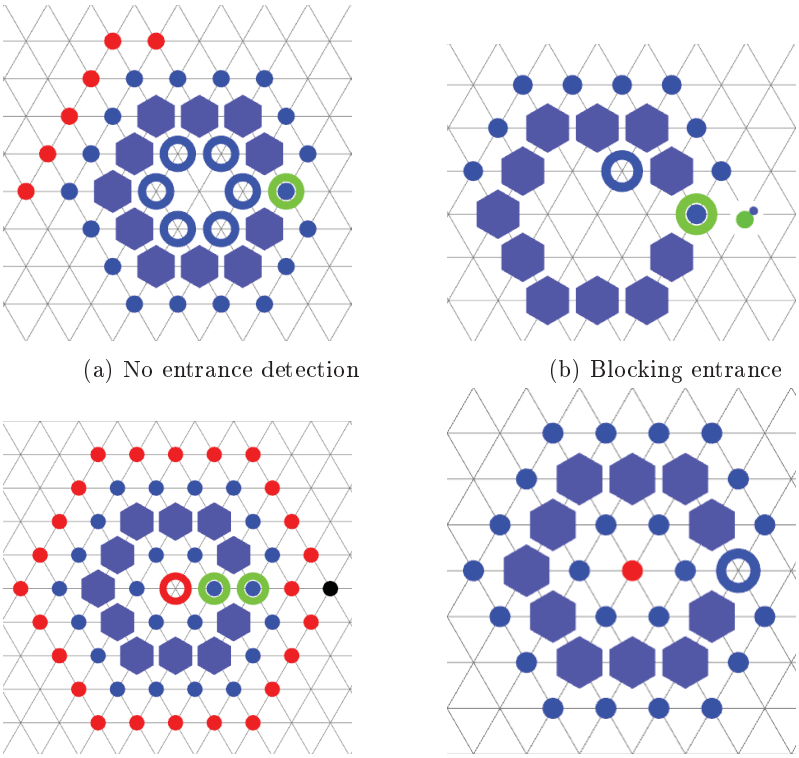


Figure 4.3: Overview: Simple Leader Based Coating Algorithm [10].

Now we use the same number of robots as before. However, the swarm must coat a cave-shaped object, as shown in Fig. 4.4a. The result of the evaluation is shown in Fig. 4.6a. As we can see, the coating was successful at 18 robots. From the 19th robot onwards, the validation becomes zero. The reason is that the leader does not recognize the cave entrance (Fig. 4.4a). It is continuously coating the outside of the cave and neglecting the inside. Therefore, the first challenge is giving the leader the ability to recognize a cave entrance.

Challenge 1: How does a leader detect an entrance of a cave-shaped object? Here, the coating is done on a trigonometry grid, and we labeled each of the six adjacent directions to help the leader find an entrance. Therefore, the leader scans the environment and labels all its adjacent locations based on the previous coming location (pr), trail location (tr), total free location(tf), free location (fr), and occupied location (oc) as shown in Fig. 4.5a. Additionally, each of these labels has a predefined unique number. The numbers must be unique and follow this $pr \ll oc \ll fr \ll tr \ll tf$ schema, which means they should be much lower than the others. So, the leader scans its environments, labels them, and then sums them up. It receives a number and compares it to a list. This list is created by defining the types of environments as shown in Fig. 4.5. We got four types of environments that have a unique number. Based on those numbers, the leader knows its environment, can act on them, and recognize an entrance.



(a) No entrance detection

(b) Blocking entrance

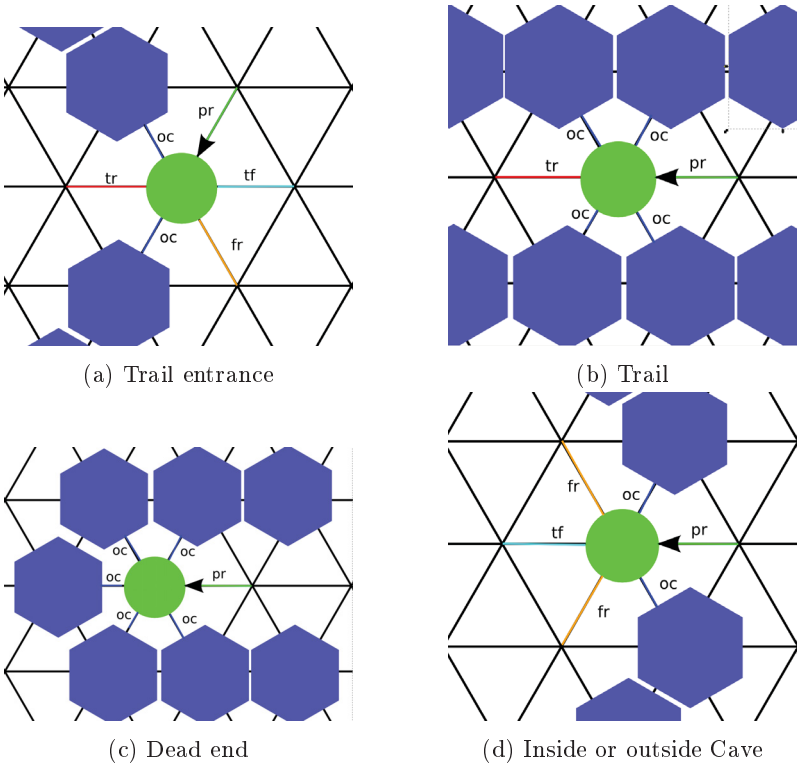
(c) No trail detection

(d) No amount control

Figure 4.4: Cave Challenges [10].

Now, the second version of the algorithm is finished. But, this time the simulator hangs without termination. Even though the leader could detect the entrance and scan the inside of the cave, the leader could not enter the cave after positioning some robots because the entrance became blocked due to the previous coating. That is the reason for the simulator's hanging because it causes the leader to stay in front of the blocked cave entrance (Fig. 4.4b). As a result, the next challenge is to define the coating order so the entrance is not blocked while coating.

Challenge 2: How to avoid the blocking of the entrance while coating? How to avoid a blocked entrance while coating? After the leader scans the object, it starts to coat the object in the order it stored the locations' coordinates, i.e., First In First Out (FIFO). However, the FIFO coating causes the cave entrance to be coated before the inside has been. Consequently, the leader cannot move further and gets stuck (Fig. 4.4b). It cannot continue the coating. Nevertheless, this challenge is straightforward to solve by changing the memory access from FIFO to Last In First Out (LIFO). Therefore, the leader reads the last stored location up to the first stored one. As a result, the inside of the cave is coated first before the cave entrance. In evaluating our second version of this algorithm, the simulator is terminated, but the success rate is not 100 percent. The validation stops after the 44th robot, and the success rate drops to zero (Fig. 4.4c).



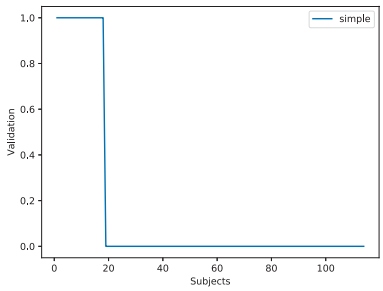
(a) Trail entrance (b) Trail

(c) Dead end (d) Inside or outside Cave

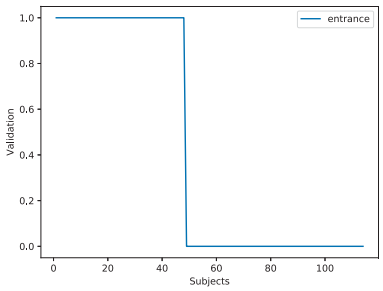
Figure 4.5: Environment and Neighborhood Types [10].

The next challenge is not as easy to explain and solve. As we can see from the result in Fig. 4.6b, the validation goes to zero after the 45th robot. An overview of this condition is shown in Fig. 4.4c. The middle, red-marked location inside the cave is not coated, even though there are enough robots. The black robot on the left is the reason that the coating becomes invalid. The black robot is the 45th robot that makes the algorithm fail. The reason is that the distance of this robot is three hops, and the distance of the uncoated, red-marked location in the middle are two hops. As a result, the coating is not valid because the distance of the free location in the middle is smaller than the distance of the black robot. So, the leader needs to know in advance to coat the middle of the cave before starting to coat the third layer, which brings us to the third challenge.

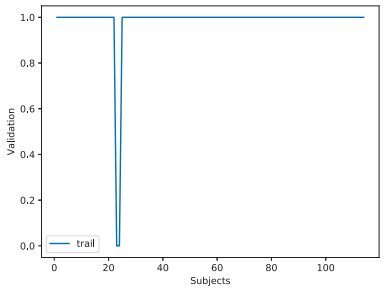
Challenge 3: How to coat the center of the cave? As we know already, the middle of the cave is not being coated because the leader is scanning and storing only the object's surroundings and is not aware of the created new center in the middle of the cave (Fig. 4.4c). As a result, the middle of the cave is not being scanned and coated, while the coating of the next layers has been started. To solve this challenge, we define the environment "trail" (Fig. 4.5b) and "trail entrance" (Fig. 4.5a). While scanning, the leader stores these environments in a separate memory. But before that, how does a leader know that there is a trail and where it ends? By adding a fourth environment termed "dead-end" (Fig. 4.5c), it helps the leader to recognize the end of a trail. While scanning, when the leader hits a "dead-end," it will



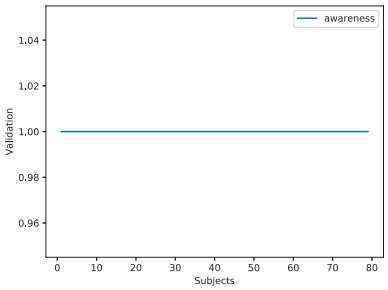
(a) Simple: Success rate 15.79%



(b) Entrance and LIFO: Success rate 42.11%



(c) Trail scanning: Success rate 98.25%



(d) Reduction: Success rate 100%

Figure 4.6: Success Rates of the various Algorithm Elements [10].

start to escape from the trail. While escaping, it starts to store each location of the trail in a separate memory. When it reaches the trail's exit (i.e., the entrance of the cave), it starts the standard scanning procedure, which means storing the location in the normal memory. After the scanning is finished, the leader starts to coat the trail from the dead-end to the cave entrance. When this has been done, it coats the surrounding of the object. With this procedure, the challenge of coating the center has been fulfilled. By testing the third version, we see in the evaluation that the success rate is now 98.25 percent (Fig. 4.6c). A tiny crinkle of invalidation can be seen at the 23rd robot, which brings us to the last and most difficult challenge.

Challenge 4: How to be aware of the number of robots and their correct position within the trail? In Fig. 4.4d we see why the validation becomes zero with the 23rd robot. The 23rd robot is the red robot in the middle of the cave. This robot's distance from the object is two hops. However, the blue marked accessible location has a distance of one hop, so the coating is invalid. The trail coating algorithm from the previous challenge fills up the cave's center without worrying that it might be unnecessary because there is an accessible location with a lower distance than the center. This problem is caused because there are enough robots to coat the object without coating the center. Because the distance of the centered free location is higher than the distance of the outside, free location. Therefore, the coating algorithm must always know the number of uncoated robots to ensure that the trail is coated correctly and

Figure 4.7: Lion mother transporting her cub (Source: [153])

validated. We have found that the number of scanned trail locations must always be even to solve this challenge. Additionally, the head and tail, i.e., the last (trail-entrance) and first (dead-end) scanned location in trail memory, must always have the same distance from the object. This information is essential to define the trail coating sequence based on the number of robots. Thus, before the trail coating starts, the leader must compare the number of scanned trail locations to the remaining number of uncoated robots. If the number of trail locations is equal to or more than the number of robots, nothing needs to be done. However, if the number of robots is less than the number of trail locations, some of the scanned trail locations must be deleted based on the number of robots. However, the order of cutting the trail is essential. The leader must delete the locations from trail memory based on the sequence: first, the trail entry, then the head, until the total number of the scanned trail locations is equal to the number of robots. With this comparison and deletion, the last problem of the leader coating algorithm is solved, and the result of the evaluation in Fig. 4.6d shows a 100 percent success rate.

With this leader coating algorithm, we show that it is possible to coat cave-shaped and simple objects. We introduced a step-by-step proof of concept and a successful algorithm. However, this algorithm does not work for oversized (i.e., more significant than the diameter of two robots) cave entrances. It also does not work for nested caves (cave in a cave) or multiple caves that are connected. The aim to coat these objects is a challenge for future work. Some more challenges are extending the algorithm beyond coating 2D to coat 3D objects and testing the algorithm with many leaders to increase the efficiency. Nevertheless, coating simple and cave-shaped objects can be done with this algorithm and with one leader.

4.1.2 Importance and Impact on Dissertation

The paper presents the first contribution C3.1 of module M3. The inspiration came from a mother animal carrying her baby (Fig. 4.7), which is also the answer to the RQ3.1. To answer the RQ3.2, the aim of this algorithm is to coat cave-shaped objects. We provide a swarm

with one leader and a cave-shaped object for coating it for this algorithm. The feature of the leader is to can scan the object and take and drop other robots, which is the answer to RQ3.3. The answer to RQ3.4 is that the leader first scans the object, calculates the number of robots needed, takes each robot one by one, and coats with them the object. To answer the RQ3.5, the leader-based coating is valid whenever the maximum distance of all the robots is smaller (or equal) to the minimum distance of all accessible locations. The coating algorithm terminates when the leader coats all the robots and itself. We use the Swarm Size as the first metric. We started with two robots and increased it by up to 120 robots each time. The second metric is the Validation. With these two metrics, we answered RQ3.6. The importance and impact on the dissertation with this paper is that all the research questions of the problem statement PS3 are solved.

4.1.3 Contribution

This paper provides a new type of coating. A leader is chosen based on its capability within the swarm. Its capabilities include scanning its environment, storing location coordinates into its memory, and taking and dropping robots. Based on our knowledge, there is no leader-based coating that has already been developed making this a main contribution. Another contribution is the coating of cave-shaped objects with our algorithm. The coating of a cave-shaped object has not been of concern in previous works. Most of the coating algorithms only considered simple objects. In summary, we provided an algorithm that uses a leader for coating, and therefore, saves effort in the production of highly intense robots, and one that is capable of coating cave-shaped objects.

4.1.4 Personal Contribution

Ahmad Reza Cheraghi's contributions to this work are include the scientific approach, problem definition, and solution for the leader coating. He has fully and independently performed the solution, design, implementation, and evaluation of the Leader-based coating algorithm. He also authored and completed the present work. Kalman Graffi was continuously involved in discussing the scientific approach of the algorithms. In addition, he supervised everything and critically revised this work.

4.2 General Coating of Arbitrary Objects Using Robot Swarms

This section summarizes the contributions and gives a verbatim copy of our paper [11].

Ahmad Reza Cheraghi, Gorden Wunderlich, Kalman Graffi
"General Coating of Arbitrary Objects Using Robot Swarms".
In: *Proceedings of IEEE 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS))*.
2020.

4.2.1 Paper Summary

Soon, nanobots might be a healing method in destroying harmful matters within the body. These bots will be injected into the body, and their mission will be to localize unhealthy cells to coat and destroy them. This procedure of localizing and coating objects, such as a cancer cell, by a robot swarm is the aim of this paper. Given are robots with countable characteristics and an object with an arbitrary shape. The characteristics of the robots are a follow:

1. The robots do not have any sensors for calculating their distances towards the objects.
2. Each robot can communicate with its adjacent neighbors.
3. They have a sensor for scanning their adjacent neighbors to find robots, objects, or free spaces.
4. There is no central unit, so no external help is provided for the coating.
5. All robots are homogeneous, meaning that they all are equal in their software and hardware.
6. Each robot has a memory for storing information necessary for the coating procedure.

The arbitrary object can be a single tile, a cave, a tunnel, a bottle, or many other things, as shown in Fig. 4.8. Therefore, the challenge for the robot swarm is to coat any of those arbitrarily shaped objects with their above-defined characteristics.

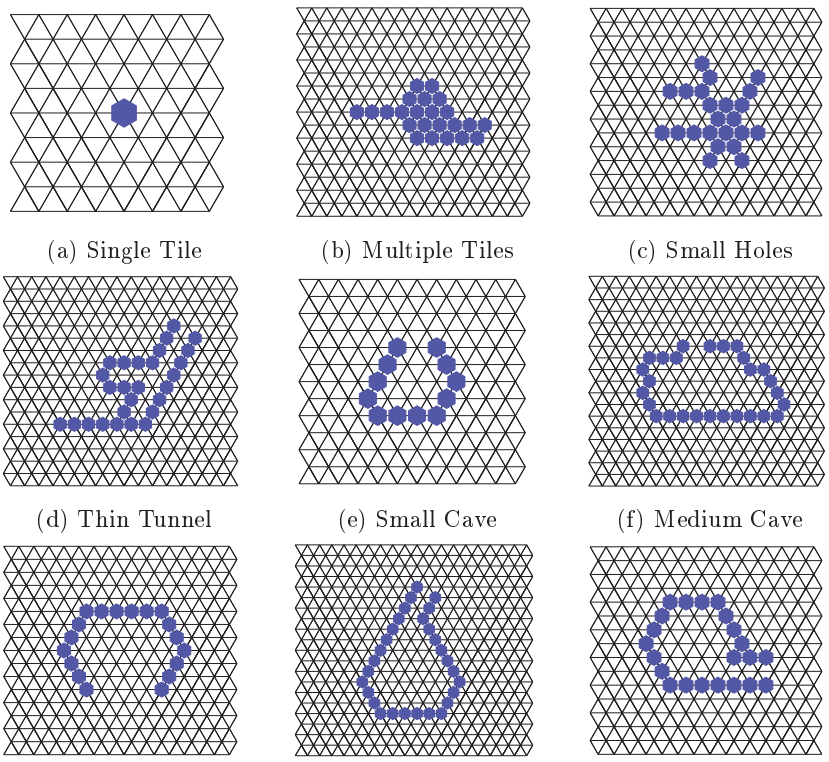
However, how can we check that the coating is successful? For a successful coating, the following condition must be true:

$$max(OwnDist[p]_{\forall\ p \in P}) \leq min(OwnDist[fl]_{\forall\ fl \in FL}) \tag{4.2}$$

This equation states that each robot p within the swarm P must coat the object as tightly as possible. Generally, the maximum distance of one of the robots within the swarm to the nearest object must be smaller or equal to the minimum distance from all the free locations fl within the set of all free locations FL. If this condition is fulfilled, then the coating is valid.

Examples of correct coatings are shown in Fig. 4.9. The blue tiles are the objects, and the red dots are the robot. In all the pictures, the robots are as tight as possible, and the distances of all free locations (the small black dots) are more significant than the distance of all robots. On the other hand, invalid coatings are shown in Fig. 4.10. The marked green locations are free locations, and their distances are lower than the robots.

The aim of this paper is to develop an algorithm for coating any arbitrarily shaped object. In Fig. 4.11 we show, with a simple example, that the general coating algorithm (GCA) works with a simple one-tile object. In Fig. 4.11a we see two robots with the initial distance of infinity and a blue-tile object with a distance set to 0. For defining the distance, both robots scan their environment first. However, the robot close to the tile gets to know that it has a robot and an object as neighbors. Therefore, that it has an object as an adjacent neighbor, its distance changes from infinity to one because it is only one hop away from the object. The other robot's distance stays to infinity. Next, the robot with a distance of one sends its distance to its neighbor robot, as it is shown with the arrow in Fig. 4.11b. Based on this information, the infinity robot's distance changes to two because it received the information from its adjacent neighbor that it has a distance of one. The robot with a distance of one now recognizes an adjacent free location with a one-hop distance and an adjacent robot with a two-hop distance. As a result, it moves towards the free location to make space for the second robot, as is shown with the big black arrow in Fig 4.11c. What happens next is that the second



(a) Single Tile (b) Multiple Tiles (c) Small Holes

(d) Thin Tunnel (e) Small Cave (f) Medium Cave

(g) Large Cave (h) Bottle (i) Small Bottle

Figure 4.8: Different arbitrary shapes [11].

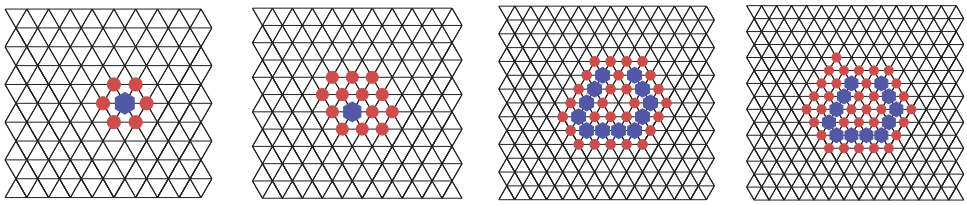


Figure 4.9: Valid coating examples [11].

robot moves to the free location occupied before by the first robot and obtain a distance of one because its adjacent neighbor is a tile. The result is shown in Fig. 4.11d. All the robots have a distance of one now and have coated the tile object.

However, this type of coating is elementary. Coating a cave-shaped object is more complicated. In Fig. 4.12a we see a red-coated ring of robots around the cave-shaped object, and in the middle of the cave, there is a green space that is uncoated. However, there is one green robot outside the coated ring, and this robot's distance is higher than the free location inside the cave. So, the challenge for the coating algorithm is to make the robots around the free space aware of the outside, green robot, thus, to fill up the green free location. But how do those robots within become aware that there is a robot with a higher distance?

For this awareness, we define the variable p_max, the maximum distance of a robot within the swarm, and set it in two ways. First, after movement and based on the distance calculation. Each robot calculates its own p_max based on its adjacent locations. For example, in Fig. 4.11b the p_max for the robot with the distance one is two because it sees either an adjacent robot or free locations that are not near the object, which, therefore, must have a distance of two. The second option is that each robot shares its p_max with its adjacent robot. It compares its own p_max value with those received from its adjacent robots, chooses the highest one, and stores it. With the last procedure, the GCA makes certain that the p_max value is shared within the swarm. Thus, the robots are permanently storing the highest distance that is within the swarm. As a result, if one of the robots around the green free location receives a p_max higher than the distance of this free location, it will move into it to make space for the others. All other robots will move as well until a space is created for the green robot outside the ring, as shown in Fig 4.12b. It then moves inside it, and the coating is valid (Fig 4.12c). This maximum particle distance, termed p_max, is necessary. Otherwise, a coating of a cave-shaped object is impossible.

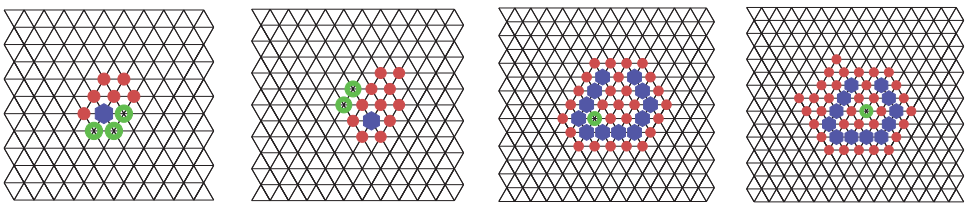
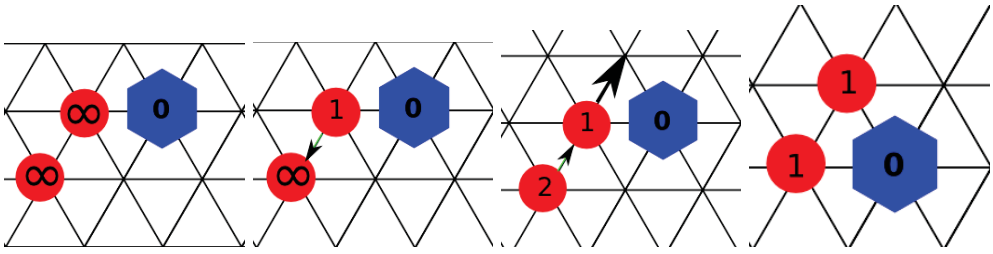


Figure 4.10: Invalid coating examples [11].

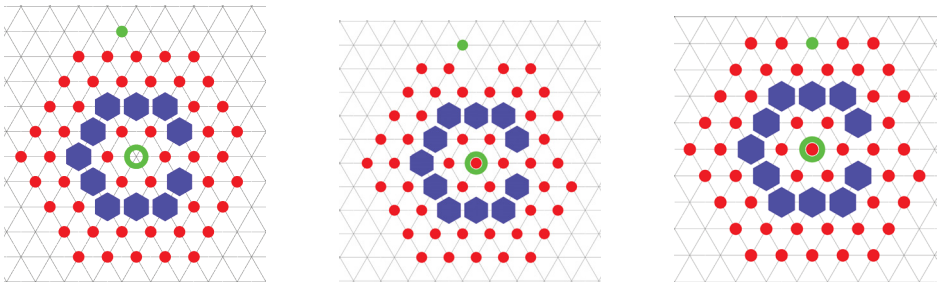


(a) No distance (b) Got distance from tile (c) Got Distance from particle and tells it to move (d) Final position

Figure 4.11: Example of simple GCA [11].

Nevertheless, sharing the p_max value brings another challenge, and this is the count to infinity problem. As we mentioned before, the p_max value is used to make other robots move. However, after that, the robots moved and made space for the one with the highest distance (i.e., p_max). However, after the coating has been done, other robots that did not move still have the old p_max. These robots distribute the p_max to others within the swarm.

Consequently, if there is still a free location with a lower distance than the p_max, those robots will move toward this location because they think there is a robot at a greater distance. As a result, the whole swarm continues to move without termination. This problem is known as the count to infinity problem. For solving this problem, each robot sends with the p_max value an additional lifetime value. This lifetime value starts with zero and increases whenever the robot thinks it is still the highest distance. However, the lifetime changes either when the robot receives a p_max value higher than its own p_max or when they are equal, but the received lifetime is higher than its own. Nevertheless, the p_max of a robot is reset when the lifetime becomes zero or after a robot's movement occurred. So, the lifetime increased by the one with the highest p_max. On the other hand, each robot receives the p_max, which decreases the lifetime by one, and shares it with others. The general coating algorithm ensures that outdated p_max values will not stay eternal within the swarm with this simple



(a) A free, empty location and a robot in an invalid coating position. (b) Entering the free location in the cave and making a new space. (c) Entering the new space

Figure 4.12: Coating the hole in the Cave [11].

Table 4.1: Runtime test overview: Table showing the minimum, average, maximum and standard deviation of the number of rounds taken for each combination of Scenario and Particle count [11].

Particle	**Scenario**											
count	**Single Tile**				**Multiple Tiles**				**Small Holes**			
	Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
20	335	388	503	30	316	393	461	32	308	382	425	29
40	371	424	518	38	347	404	533	35	395	545	830	83
60	446	597	832	89	401	639	923	104	379	456	602	54
80	479	634	875	88	425	546	755	81	434	618	1058	126
100	521	676	926	85	533	795	1157	143	461	614	833	86
	Thin Tunnel				**Small Cave**				**Medium Cave**			
	Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
20	302	388	485	31	332	418	542	34	296	377	455	33
40	403	527	740	64	377	446	524	38	466	600	782	80
60	560	681	836	61	425	512	647	56	1079	1260	1559	97
80	545	680	1007	79	461	611	902	85	914	1031	1211	73
100	557	705	983	84	467	616	875	94	1118	1415	1802	139
	Large Cave				**Bottle**				**Small Bottle**			
	Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
20	311	388	461	29	323	384	455	30	308	383	455	31
40	508	720	941	81	457	594	884	77	692	856	1061	88
60	464	545	671	46	1370	1592	1850	106	686	782	944	55
80	626	884	1267	126	1136	1271	1523	71	896	1200	1619	150
100	674	906	1205	122	1231	1422	1715	117	1046	1276	1649	135

solution. The information propagation method is needed to share information such as one's own distance and the maximum distance (p_max) of any robot in the swarm. First, each robot receives the information from its adjacent neighbors and processes it. Next, the robots send the processed data to its adjacent neighbors. The robots use this method to communicate and based on the received information, they either update their distances or start to move. With this information propagation, GCA ensures that all the information regarding distances and the p_max within the swarm is updated. However, sometimes the highest distances reach robots earlier than the lower distances. Thus, the robots receive the wrong distances after each movement. To solve this problem, GCA lets each robot, after a movement, wait for one round, so all the other robots can move and update each, self-based on the new position. Afterward, all the robots start to send messages. Now that we know how the algorithm works, let us have a look at the evaluation.

For the evaluation, we use Swarm-Sim. We use nine different arbitrary objects as shown in Fig.4.8. The robot swarm size varies between 20-100 robots, with an increase of 20 robots after each simulation. The robots are located randomly in the Swarm-Sim world. To make sure that the moving and choosing of the robots happen randomly each time, we use 20 different seeds for each setup. In total, we did 900 tests. The maximum round number is 10,000. If the swarm cannot carry out a valid coating within 10,000 rounds, the simulation is noted as failed. The result of our evaluation is shown in Table 4.1. As we can see, all the simulations

Table 4.2: Runtime test overview of the robustness test: Table showing the minimum, average, maximum and standard deviation of the number of rounds taken for each combination of Scenario and Particle count [11].

Particle	**Scenario**											
count	**Single Tile**				**Multiple Tiles**				**Small Holes**			
	Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
20	317	395	493	36	300	387	449	38	300	380	437	34
40	365	435	545	41	359	395	461	24	407	494	701	56
60	410	522	746	68	415	530	656	60	386	439	511	32
80	449	587	833	81	410	513	683	60	437	635	905	95
100	503	680	890	88	530	681	941	93	448	581	866	85
	Thin Tunnel				**Small Cave**				**Medium Cave**			
	Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
20	287	384	461	34	322	414	499	36	295	374	446	35
40	398	499	703	63	377	453	560	43	428	551	748	76
60	578	696	836	60	413	539	695	65	1081	1295	1570	119
80	509	660	857	68	440	621	932	112	929	1066	1229	75
100	599	745	959	89	410	607	827	88	860	1144	1490	131
	Large Cave				**Bottle**				**Small Bottle**			
	Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
20	310	385	454	28	320	376	446	30	300	381	467	36
40	508	669	884	87	446	555	745	75	523	761	959	110
60	443	555	698	57	1022	1555	1880	157	707	806	1073	68
80	526	772	1010	118	1109	1312	1499	89	725	905	1240	124
100	728	949	1400	133	1010	1242	1478	107	1145	1351	1736	132

have an average round below 10,000 rounds. As a result, the success rate of the general coating algorithm with nine different arbitrary-shaped objects is 100 percent.

Additionally, the general coating algorithm must fulfill the robot swarm criteria based on [27]] i.e., robustness, flexibility, and scalability. Robustness means that if some of the robots are missing or lost, the algorithm must still solve its task. A flexible algorithm must adapt itself to changes in the environment, and a scalable swarm algorithm must deal with increases or decreases of robots within the swarm.

We want test if the GCA can handle a larger number of robots in the swarm for scalable testing. We will use the previous setup with 120 to 500 robots and increase the swarm by adding 20 more robots after each simulation. The result is that the success rate is still 100 percent, which means that the swarm sizes could coat each of the nine objects within 10,000 rounds. Consequently, our coating algorithm is scalable.

Robustness means that the algorithm is capable of adapting itself after the loss of robots in the swarm. In order to test this, we used the same setup as before. However, instead of having more robots this time, we deleted robots while the coating simulation was in progress. The conclusion is that the running time was not affected much and all the tests ended successfully within 10,000 rounds.

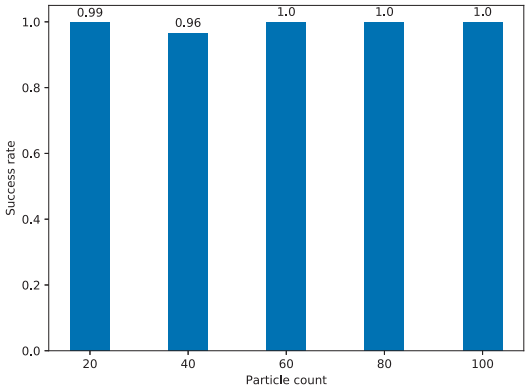


Figure 4.13: Plot showing the success rate in the Scenario with two tile objects [11].

For a swarm to be flexible the algorithm must be able to handle changes in the environment. For this reason, we simulated a scenario with two objects beside each other. We put two objects tiles close to each other, with the intention of evaluating if the swarm can coat both objects successfully. The result of the evaluation is shown in Fig. 4.13. We see that all the robots are as tight as possible around both objects and the coating is valid. In conclusion, this general coating is flexible, too.

We have shown a general coating algorithm that is capable of coating arbitrarily shaped objects. This general algorithm fulfills the criteria of the robot swarm, i.e., it is flexible, scalable, robust. Because the robots are limited, equipped with scanning environments and communication, it is cheaper and easier to produce them. The algorithm does not need any GPS or distance measurement. Therefore, the production costs of the robots can be reduced.

Nevertheless, this algorithm has some limitations. First, it is only tested on triangular grids that are limited by their directions. In the future, it would be helpful to adapt the general algorithm to environments without any grids. Another point is the simulation in 3D. If coating could be done in reality, it would be interesting to test GCA in 3D. Next, testing the flexibility should be extended with more objects and with more considerable distances.

Further, in the testing of robustness, we simply deleted the robots. However, in real-life, robots cannot disappear when they are broken. They are, instead, disabled and stuck in their position. Therefore, it is necessary to adapt GCA for when robots are not moving or communicating. In conclusion, the general coating algorithm is the first step to let swarms coat arbitrarily shaped objects, and hopefully, this algorithm can be used in the future for nanobots to detect, coat, and destroy cancer cells.

Figure 4.14: Wolves are surrounding their prey (Source: [154])

4.2.2 Importance and Impact on Dissertation

The general coating is the second contribution C3.2 of module M3. Here we provide a coating algorithm that can coat any arbitrarily shaped object. The inspiration for it comes from wolves surrounding their prey (Fig. 4.14) and as mentioned earlier in chapter 1.1 by antibodies attacking viruses (Fig. 1.1), which is the answer to RQ3.1. The answer to RQ3.2 is that this algorithm aims to coat any object with a swarm of robots. A swarm of robots has been created that can communicate and move; and that can coat an arbitrarily shaped object, which is the answer to RQ3.3. The answer to (RQ3.4) is that the algorithm works as follows. When a swarm robot hits an object, it becomes the distance one. Next, it checks for free locations and moves toward one if it recognizes a neighbor robot at a higher distance than its own. Additionally, the robots share their distances and store the maximum received distance. Therefore, if a robot receives a maximum distance higher than one of its neighbor locations, it will move towards one of them. Based on these criteria, the robots complete this coating. The coating is validated when all the robots are positioned, so the maximum distance of all the robots is smaller (or equal) to the minimum distance of all accessible locations closed, which is the answer to RQ3.5. To answer the RQ3.6, the metrics include the Rounds, Swarm Sizes, and the Success Rate.

4.2.3 Contribution

This paper aims to develop an algorithm for robot swarms to coat arbitrary-shaped objects. The main contribution of this paper is that this, proposed, general coating algorithm (GCA) works for any arbitrary object. Previous related works have only been focused on a simple object. This means they have no entrances or tails that lead inside the object. These kinds of complex objects are cave- or bottle-neck-shaped. With GCA, we developed an algorithm

that can coat any arbitrary-shaped object. To sum up, GCA is a unique algorithm for robots with a minimum amount of properties, such as communicating and scanning their adjacent neighbors to let them coat any arbitrary-shape object. Additionally, GCA is a unique algorithm contribution in accordance with the robot swarm criteria (flexible, robust, scalable).

4.2.4 Personal Contribution

Ahmad Reza Cheraghi's contributions include the scientific approach, problem definition, and solution of the general coating algorithm for arbitrarily shaped objects. He designed and developed this algorithm and structured, authored, and completed this work. Under his supervision, Gorden Wunderlich implemented and evaluated the general coating algorithm and wrote some parts of this paper. In addition, the solution was refined in several discussions with Ahmad Reza Cheraghi and Gorden Wunderlich. Kalman Graffi provided the initial idea and design of the general coating algorithm. He also supervised the entire project and critically reviewed this work.

Chapter 5

Challenges and Possibilities of Communication within a Robot Swarm

Communication is key. The same goes for robot swarms. The robot within the swarm needs to communicate to share information about itself or the environment. However, how can they communicate without having any overhead? What type of communication should they use? Should they communicate directly or indirectly? This chapter answers the above questions. We consider both types of communication. Direct communication uses either Wifi or Bluetooth. On the other hand, indirect communication works by sharing information through the environment. However, indirect and direct communication brings challenges, which is considered here.

5.1 Opportunistic Network Behavior in a Swarm: Passing Messages to Destination

This section summarizes the contributions and gives a verbatim copy of our paper [12].

Ahmad Reza Cheraghi, Julian Zenz, Kalman Graffi
"Opportunistic Network Behavior in a Swarm: Passing Messages to Destination"
In: *Proceedings of IEEE 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS))*. 2020.

5.1.1 Paper Summary

Transferring data over the Internet is standard nowadays, and everyone relies on the infrastructure of the Internet. However, what happens if the Internet is shut down? A shutdown happened ten years ago in Egypt and more recently in Iran. In Egypt, with the revolution, the government decided to shut down the Internet from the entire world to stop the demonstration organized chiefly over social media sites such as Facebook. The same tactic was used by the Iranian government two years ago. By shutting down the Internet (usually done by shutting down the ISP or major Internet backbones), sharing information through the traditional

client-server mechanism is impossible. This is why a new network has been proposed termed opportunistic networks.

Today, almost everyone uses a smartphone, and each one is equipped with Wifi and Bluetooth. With the help of one of these systems, it is possible to transfer data between smartphones without using the Internet. Each smartphone can act as a router to transfer messages from the sender to its receiver, this kind of network is termed an Opportunistic Network (OppNet). OppNet is still an interesting and exciting research area, and plenty of work has been done on it [155, 156, 157, 158, 159, 160].

An OppNet is a subcategory of a Tolerant Delay Network (DTN). With DTN, the sender and receiver tolerate delays within the network. Typically, DTN is used for long-distance communication, e.g., between a Mars rover and earth. However, with OppNet, the delay toleration occurs because the link between the sender and receiver is established through other nodes (such as smartphones) that are mobile and, therefore, do not have a fixed position. It could be that there is not a connection between the sender and receiver. Additionally, all the nodes must apply the store, carry, and forward principle to build an OppNet. Consequently, they must act as a router, and therefore, new routing protocols must be defined to make the OppNet more efficient. For this paper we chose the Epidemic routing and PRoPHET.

The Epidemic routing is straightforward. As the name says, data is spread like an epidemic to any node within the communication range. Therefore, it maximizes the chances of data delivery. On the other hand, however, it increases data overhead and storage. Overhead means that the network itself must handle the high amount of data transfer, which sometimes leads to networks failure. Additionally, each node needs to store each received message until it can be forwarded to another node. Therefore, the demand for high storage is mandatory.

On the other hand, the probability routing protocol, using history of encounters and transitivity (PRoPHET), not only sends data whenever a node is encountered, but its principal idea is to reduce the overhead by adding delivery probabilities. PRoPHET shares data depending on calculated probability. Each node starts with an initialized delivery probability, and it changes depending on the last time it encounters another node, especially when it is a receiver. For data forwarding, each node checks (by encountering another node) if the delivery probability of the other node is higher than its own, and if so, it will forward the message. Thus, PRoPHET saves overhead and storage. Before the comparison of Epidemic with PRoPHET starts, we define the metrics.

This paper aims to compare the OppNet routing protocols Epidemic and PRoPHET in SwarmSim to test the compatibility of the OppNet with swarms. For this paper, the particles within a swarm can move within a given mobility pattern and send messages after a defined time.

In this paper, we simulate a swarm with 200 up to 600 robots and evaluate them with the following metrics:

In this paper, we simulate a swarm with 200 up to 600 robots and evaluate them with the following metrics:

1. Delivery Success: the rate of data delivered (without considering duplicates) to the receiver.

Table 5.1: Particle parameters used in the experimental setup [12].

Parameter name	Parameter values
Mobility model structure	{random-walk, random-circles}
Mobility model steps	[10, 30]
Scan radius	{5, 10}

Table 5.2: World and solution parameters used in the experimental setup [12].

Parameter name	Parameter values
(max_x, max_y)	(200, 200)
Initial particle placement	random
Particle count	{200, 400, 600}
New message interval	Every 5th round
Max round	1000

2. Delivery Efficiency: average delivery rate of duplicates that the receiver receives. The closer this value goes to one, the fewer duplications of data have been delivered to the receiver, which means that the routing protocol avoids forwarding replications.

3. Overhead: the average number of replicas per generated data. This number is the amount of forwarded (replicated) data to other particles.

We use the Swarm-Sim with the setup shown in Table 5.1 and 5.2 for the simulation. The given parameters in the table show which ability each particle has. Each particle is equipped with two types of mobility models. The first model is a random walk, which means that each particle chooses its direction randomly and moves in this direction for either 10 or 30 steps. After finishing, it chooses a random direction, and the procedure repeats. The second mobility model is the random circle walk. In this mobility model, the particle chooses a direction again randomly. However, it does not go straight. It makes a clockwise circle move. When it hits its starting point, it randomly chooses another direction and proceeds with the same routine. The second parameter, (mobility model steps) in Table 5.1, defines the number of steps a particle should go. The last parameter is the scan radius, which defines the communication range for transferring data between the particles. The scan radius is either five or ten hops. Table 5.2 shows the setup of the simulator's scenario (the playing ground). The world size is 200 times 200 locations. A location is the coordinates point on the Swarm-Sim trigonometry grid, and the distance between two locations is termed one hop. All the particles are located randomly in this world. We test each routing protocol with 200 and 400 particles. Every fifth round, a new message is generated and sent. The maximum round number for termination is 1000.

The results of the simulation are shown in Table 5.3, 5.4,and 5.5. The first columns show the mobility model, particle number, and communication radius. The second and last columns stand for the Epidemic and PRoPHET routing algorithm result.

The first comparison of these two routing algorithms is for delivery success (Table 5.3). At

Table 5.3: Delivery Success for Epidemic and PRoPHET Routing [12].

Parameters	Epidemic	PRoPHET
{random-walk, 200, 5}	0.150	0.020
{random-walk, 200, 10}	0.175	0.055
{random-walk, 400, 5}	0.230	0.055
{random-walk, 400, 10}	0.115	0.130
{random-walk, 600, 5}	0.385	0.065
{random-walk, 600, 10}	0.085	0.135
{random-circle, 200, 5}	0.155	0.020
{random-circle, 200, 10}	0.165	0.090
{random-circle, 400, 5}	0.625	0.250
{random-circle, 400, 10}	0.120	0.055
{random-circle, 600, 5}	0.700	0.140
{random-circle, 600, 10}	0.275	0.130

Table 5.4: Delivery Efficiency for Epidemic and PRoPHET Routing [12].

Parameters	Epidemic	PRoPHET
{random-walk, 200, 5}	0.395	1.000
{random-walk, 200, 10}	0.124	1.000
{random-walk, 400, 5}	0.619	1.000
{random-walk, 400, 10}	0.319	1.000
{random-walk, 600, 5}	0.526	1.000
{random-walk, 600, 10}	0.230	0.400
{random-circle, 200, 5}	0.155	1.000
{random-circle, 200, 10}	0.686	1.000
{random-circle, 400, 5}	0.351	1.000
{random-circle, 400, 10}	0.120	0.491
{random-circle, 600, 5}	0.391	0.867
{random-circle, 600, 10}	0.037	0.403

first, we see the Epidemic routing does better overall compared to PRoPHET. Only in two cases, the PRoPHET success rate is higher, i.e., with the particle amount of 400 and 600 with random-walk and a communication radius of 10. Hence, a smaller communication range means less interaction. Nevertheless, the reason for this result needs further examination. As a result, the delivery success rate of Epidemic routing is superior to PRoPHET in the majority of simulations.

The following Table 5.4 shows the results of the delivery efficiency. Now we see that here ProPHET surpasses Epidemic. Although Epidemic has higher delivery success, the efficiency is lower. This means more data is delivered. However, this data is duplicated. Therefore, the efficiency decreases. ProPHET probabilistic sharing lowers the creation or duplication.

The last comparison is the overhead. The overhead is a metric that says how many duplications have been generated from the real generated messages. The higher the overhead value, the worse the routing algorithm because a routing algorithm should reduce overhead. Otherwise, the receivers go down for receiving massive amounts of data and the network becomes congested

Table 5.5: Overhead for Epidemic and PRoPHET Routing [12].

Parameters	Epidemic	PRoPHET
{random-walk, 200, 5}	318.565	112.265
{random-walk, 200, 10}	1225.945	510.83
{random-walk, 400, 5}	1324.385	479.035
{random-walk, 400, 10}	4783.535	1955.125
{random-walk, 600, 5}	2979.19	1122.46
{random-walk, 600, 10}	10341.215	4827.07
{random-circle, 200, 5}	344.58	112.04
{random-circle, 200, 10}	1302.504	590.295
{random-circle, 400, 5}	1371.465	495.205
{random-circle, 400, 10}	6476.815	2299.34
{random-circle, 600, 5}	3026.25	1170.275
{random-circle, 600, 10}	11448.555	5089.92

with duplicate messages. The result of the overhead is shown in Table 5.5. As expected, the Epidemic overhead is much worse than PRoPHET regardless of the communication radius, mobility, and particle amount. It is over two times worse than ProPHETs overhead. Regardless, we also see that for both routing algorithms, the overhead increases as the communication radius changes from 5 to 10 hops.

This paper compares two OppNet routing algorithms, i.e., Epidemic and PRoPHET, in a swarm environment and the new simulator Swarm-Sim. Swarms of particles have been evaluated against each other with different numbers, routing protocols, mobility models, and communication ranges. The results show that the Epidemic routing is superior in delivery ratio. Based on the cost of low efficiency and high overhead. In contrast, PRoPHET has a lower delivery ratio but is highly efficient and has a lower overhead.

Future work must be done in comparing these results with the OppNet simulator ONE [161]. Adding more OppNet routing protocols for comparison and more metrics, such as message delay or memory consumption will be interesting. In general, in this paper, we have proven that the Swarm-Sim can simulate an OppNet making this simulator ready for implementing and testing more exciting networks.

5.1.2 Importance and Impact on Dissertation

With this paper, we present for module M4 the contribution C4.1. This paper elaborates on direct swarm communication using opportunistic networks (OppNet). To answer RQ3.1, the inspiration comes from our human nature, and that is gossiping. We use gossiping as a kind of store, carry, and forward transportation of rumors (Fig. 5.1). However, this paper aims first to adopt two OppNet routing protocols, i.e., Epidemic and ProPHET, on the Swarm Simulator and compare them to each other, which is the answer to RQ3.2. The answer to RQ3.3 is that given are robots with the feature to move and communicate wireless. We answer the question RQ3.4 as follows. The robots share data whenever they are in their connection range. However, the data is shared on every connection by the Epidemic routing. The data is flooded within

Figure 5.1: Businesspeople Gossiping (Source: [162])

the swarm network. However, the ProPHET routing protocol does not flood the network and chooses forwarding based on the previous encounter with whom it should forward the data. Each agent calculates delivery probabilities depending on the last encounter. Encountering an agent increases the delivery probability for the encountered agent and influences other agents' probabilities. The delivery probability for the receiver of a message an agent carries decreases over time. The aim is reached when the sent data reaches its destination, and this is the answer to RQ3.5. The answer to RQ3.6 is that we use three metrics for the evaluation. Delivery Success is the rate of messages delivered successfully to the receiver without considering duplications. The second metric is Delivery Efficiency, which is the average number of duplicates delivered. The closer this value is to one, the fewer duplicate deliveries. Finally is the Overhead. Overhead is the average number of generated replicas per unique message.

5.1.3 Contribution

There are a few swarm simulators on the market. However, none of them provide any communication models for a swarm. This paper provides the first swarm simulator that uses a decentralized communication method called Opportunistic Networks (OppNet). We implemented two routing protocols Epidemic and PRoPHET, and two types of mobility models and compared both routing algorithms to each other with a swarm size of 200, 400, and 600 particles. As far as we know, no simulator uses the OppNet or any other communication methods with different mobility models for a swarm evaluation tool. Therefore, the contribution of this paper is the possibility to simulate OppNet with the swarm simulator Swarm-Sim.

5.1.4 Personal Contribution

Ahmad Reza Cheraghi's contributions to this work include the scientific approach, problem definition, and solution to using OppNet with Flooding and ProPHET routing protocols for direct communication between units within the swarm. He also wrote and finalized most parts of the thesis. Under his supervision, Julian Zenz implemented and evaluated the code. He also authored the first version of this work. In addition, the solution was refined in several discussions with Ahmad Reza Cheraghi and Julian Zenz. Kalman Graffi was continuously involved in discussing the scientific approach and provided a critical revision of this work.

5.2 Prevention of Ant Mills in Pheromone-Based Search Algorithm for Robot Swarms

This section summarizes the contributions and gives a verbatim copy of our paper [13].

Ahmad Reza Cheraghi, Jochen Peters, Kalman Graffi
"Prevention of Ant Mills in Pheromone-Based Search Algorithm for Robot Swarms"
In: *Proceedings of IEEE IRCE 2020: The 3rd International Conference of Intelligent Robotic and Control Engineering.* 2020.

5.2.1 Paper Summary

Natural phenomena are inspirational sources for computer scientists, and one example is ant foraging. While ants are searching for food, they walk without a standard pattern. However, whenever they find a food source, they spread pheromones on their way back to the nest. These pheromones help other ants find a food source quicker because the pheromone path leads them to the found food source. However, sometimes this pheromone trail leads the ants to their death.

When many ants find food sources, they will spread pheromones on their way back to their nest, which causes many pheromone paths. However, sometimes these different paths overlap each other. As a result, ant mills are created or the so-called "spiral of death." This ant mill will cause the ants to run in a circle, and instead of reaching a food source, they will die. The ants will continuously walk in search of food until they die of starvation.

This paper introduces an algorithm to prevent ant robots from building or getting into ant-mills. Given are robots, food sources, and a nest. The robots can move randomly and are equipped with a gland to spray pheromones, hunger level, and age. They can also store the path while searching for food to find back home. Each food source has a random amount of nutritional value, which decreases each time an ant robot finds it. The nest is the starting- and endpoint of each ant. The aim is to develop an ant foraging algorithm with pheromones that can prevent the creation of ant-mills. The procedure is first to develop a simple foraging algorithm, and then upgrade it with an ant-mill prevention system.

For the foraging algorithm, some steps need to be defined:

1. The ant leaves its nest and searches, in a randomly chosen direction, for food.

2. When it finds a food source, it takes a portion of it and walks back to its nest.

3. On its way back, it spreads pheromones to indirectly communicate the location of the food source. Consequently, if any other ant hits this pheromone, it will follow this path until it reaches a food source.

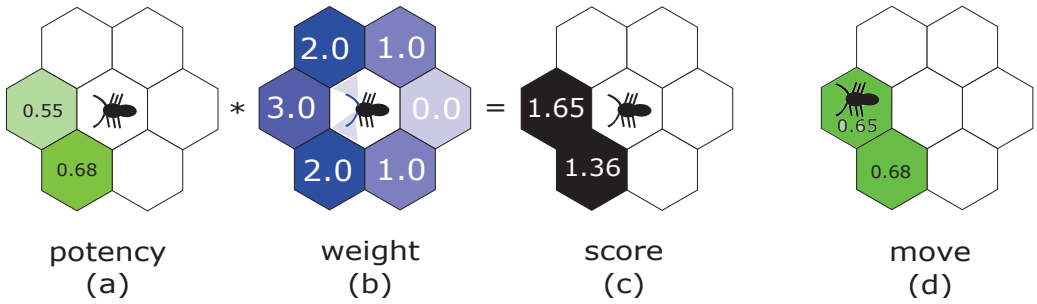


Figure 5.2: Pheromone Scoring [13].

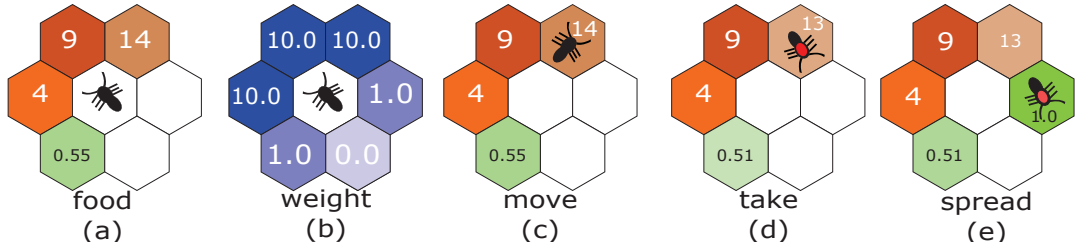


Figure 5.3: Food Scoring [13].

4. If an ant with food reaches its nest, it will leave the food and start the procedure of searching again either by randomly walking or following the pheromone path.

However, if the ant faces many pheromone paths, as shown in Fig. 5.2, it chooses one of them based on a scoring algorithm. The scoring works as follows. As we can see in Fig. 5.2a, the ant is facing two pheromone paths with different potency. The potency of the pheromone indicates how far a food source is. The ant multiplies the potency with a weight calculated based on the ant viewing direction and adjacent neighbors. The ant stores each scoring result and then chooses the one with the highest point, as shown in Fig. 5.2c. It then moves towards this pheromone trail (Fig. 5.2d). When facing a food source, the scoring algorithm is slightly different (Fig.5.3) because the weighing of the food sources is constant, already set to 10, to neglect any pheromone paths. Therefore, the ant neglects all the pheromones, multiplies the weight value of the food with the nutrition value, and chooses the one with the highest scoring value. Nevertheless, this simple ant foraging algorithm leads us to the same problem that real ants face: the ant-mill. For that, we introduce a solution to prevent ants from going into an ant-mill.

Each ant is equipped with a sensor to scan its neighbor's position for other ants. The sensor helps the ant to check how many ants are in its neighborhood. If an ant recognizes more than two robot ants in its neighborhood, the probability of getting into an ant-mill seems high. Therefore, it will randomly choose an ant-free direction and follow this direction for five steps. We termed these steps as an escape. After escaping, the ant will continue its task, i.e., either foraging or returning to the nest. This procedure prevents the ants from going into a spiral of death, which we will prove with our evaluation.

For the evaluation, we use the Swarm-Sim with the following set-up. Thirty ant robots with a hunger level of 50 rounds. The hunger level is reduced by one after each round, while the ant is searching for food. If it does not find any food within these 50 rounds, it is reborn again and starts from the foraging procedure from its nest. Next, sixteen food sources are positioned randomly, and each has a nutritional value chosen randomly between 1-20 pieces. The pheromone potency is 100 percent, and it has an evaporation rate of 4 percent, which means this rate will reduce the pheromone potency after each simulation round.

For comparison, we use three different ant foraging algorithms. The first one, the ant swarm, walks only randomly (RW) and without spreading pheromones. Second, a pheromone-based search algorithm (PSA) includes random walking and the usage of pheromones. This means that the ants spread pheromones after finding a food source. Last, the pheromone-based search algorithm is an upgrade with the ant-mill prevention (PSAMP) system. We run for 20 simulations with 20 different seeds. We use the different seeds because of the ants' random walking, food source position, and nutritional value because each different seed value brings new random values. The simulation terminates after 1,000 rounds.

For the evaluation, we use the average age and food delivery as metrics. The average age shows how old all the ants, on average, become within the 1,000 rounds. At the simulation start, the age is zero, and after each round, it is increased by one. An ant can live a maximum of 1,000 rounds because the simulator terminates after 1,000 rounds. This age can only be reached if the ants find a food source before they starve.

On the other hand, food delivery shows how much nutrition the ants deliver from the food source to their nest. The food delivery metric is an indication of the success of the foraging algorithm. The unit of the food delivery is in percentage. It indicates the amount of food nutrition delivered to the nest. The higher the food delivery is, the better the foraging algorithm. In contrast, a lower food delivery indicates an unsatisfied foraging algorithm.

For the evaluation, each metric is considered with the three, previously defined, foraging algorithms. First, we compare the average age. The results are shown in Fig 5.4. From the beginning, the average age for all three algorithms rises equally. Suddenly, at the 50th round, the average age for the entire algorithms falls. The reason is because most of the ants died; the hunger level has reached zero. Nevertheless, from the 50th round onwards, the changes in the average age of each foraging algorithm differ from each other. For RW foraging, the average age increases from the 24th to 60th rounds between the 50th and 100th simulation round. Afterward, the average age starts to fluctuate till the end of the simulation. However, this fluctuation is not symmetrical due to the ants random walking.

Looking at the PSA average age line, we see that after the 50th round a rapid increase in the average age to almost 120 rounds. This increase occurred because the ant robots were using pheromones, and therefore, other ants found the food source easily. However, after the 200th round, the average age decreased and became lower than the RW average age. The reason is that most of the food sources have been found. Thus, the ants have created many pheromone trails, which have created ant-mills and brought the ants into the spiral of death. From the 350th round to the end, a stable fluctuation of the average age occurs, which shows ant-mill occurrences; the ants die and reincarnate in a constant ratio, i.e., after 50 rounds.

The latest algorithm in the evaluation is the PSAMP. Here, we can see from the 50th to the

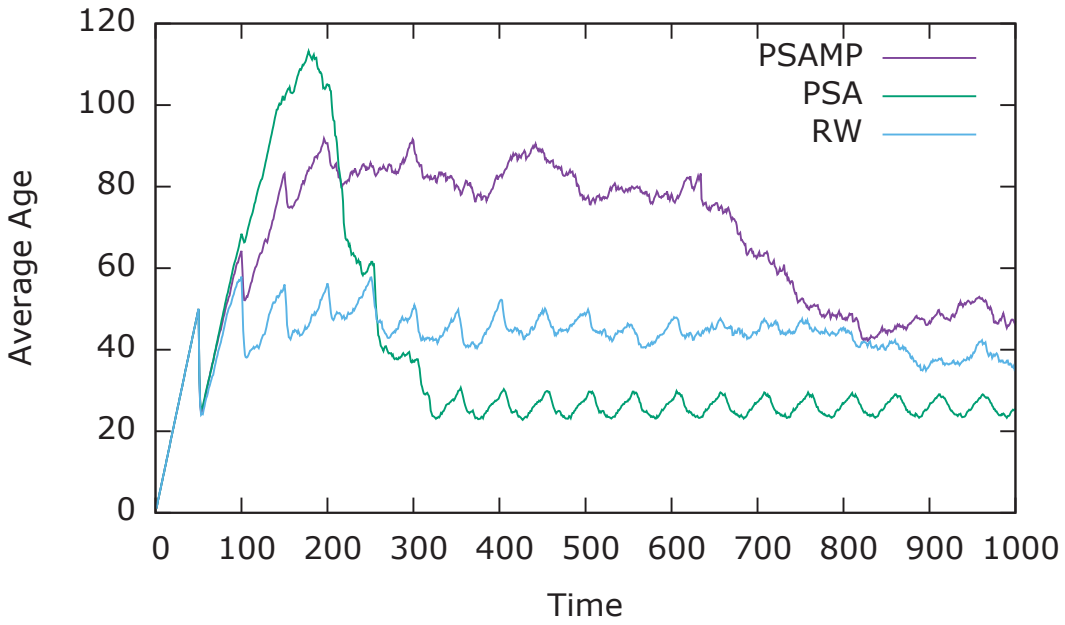


Figure 5.4: Average age with an evaporation of 4% [13].

200th round that there is an increase in the average age. Afterward, compared to PSA, there is no sudden decrease. The average age fluctuates between 70 and 90 rounds. However, from the 680th round onwards, the average age starts to decrease. This decrease is because most of the food sources have been found, delivered, and therefore finished. In general, the PSAMP average is higher compared to RW and PSA. Let us move on to the last metric, food delivery.

The result of food delivery is shown in Fig. 5.5. Up to the 50th round, the increase of food delivery for all three algorithms is equal. The RW food delivery tends from this point onwards to become slower than the others, but it still increases until the end of the simulation. The food delivery ratio of PSA increases faster but becomes constant from the 150th round. The PSA food delivery seems to be stopped, which indicates that the ants are stuck in the spiral of death. As a result, the RW foraging algorithm is more efficient than the PSA, with no ant-mill prevention. In contrast, the PSAMP food delivery increases continuously and overtakes all others. The conclusion is that the PSAMP foraging algorithm avoids the ant-mill, and thus increasing the efficiency for collecting food and the ants' average age. However, having only a foraging algorithm with pheromones, without any ant-mill prevention system, lowers the efficiency of food delivery and the average age.

Nevertheless, the weakness of PSAMP is that it only avoids ant-mills but cannot detect or destroy them. Future work could improve the ant robots to detect ant-mills, such as counting how many times an ant visited the exact location in a given period—or destroying ant-mills by using different pheromones. Another future work could be to use different movement models to how they affect the foraging procedure of the ants.

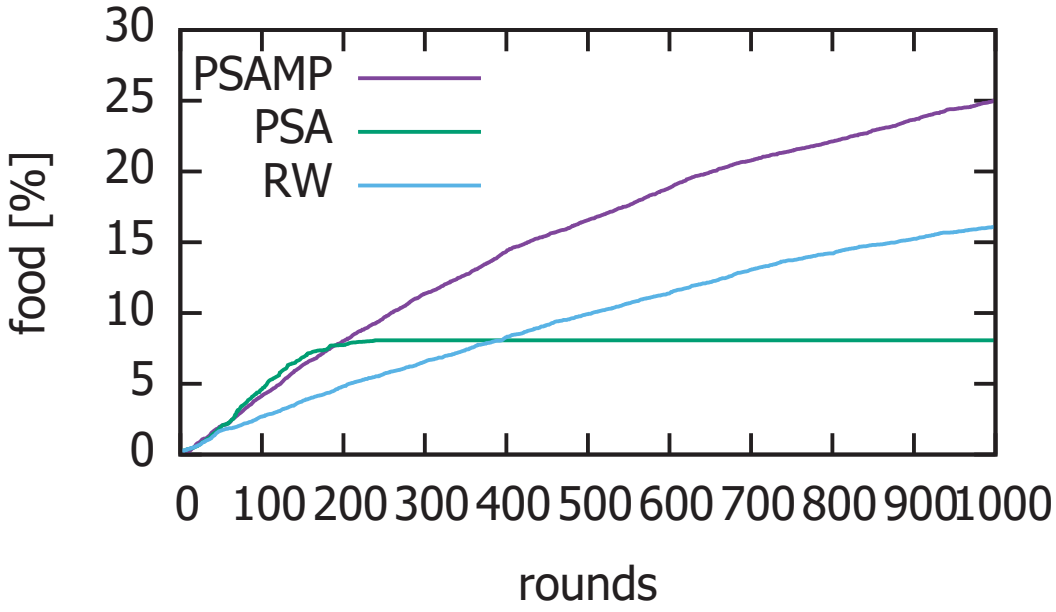


Figure 5.5: Carried food with an evaporation of 4% [13].

5.2.2 Importance and Impact on Dissertation

This paper covers the communication module M4 and contribution C4.2, an algorithm for indirect communication within robot swarms. The answer to RQ3.1 is that the inspiration for this contribution is coming from ants get stuck into the spiral of death (Fig. 5.6) and as mentioned earlier in chapter 1.1 by ants following a pheromone trail (Fig. 1.2) . This algorithm aims for ants to collect found food into their nest by preventing them from creating ant-mills, which is the answer to RQ3.2. The answer to RQ3.3 is ants, several food sources, and the nest where the ants start and end the foraging. The ants can move, take and drop food and sense neighbors. They have a gland to spray pheromones and memorize their visited path, which is necessary to find the way back home. Additionally, each ant has a pre-defined hunger level, which reduces when it becomes zero after each round. The answer of RQ3.4 is as follows. First, the ants come out from their nest and start walking randomly to find food. Next, after they find a food source, they take a piece and go back to their nest. However, we use pheromones and an ant-mill prevention system to make the algorithm more efficient. Whenever all the food sources are collected for the nest, the aim is reached. Therefore, the validation occurs when the food quantity in the nest equals the food source provided at the beginning, which is the answer to RQ3.5. The metrics include Food Delivery and Average Age, which is the answer to RQ3.6. Food Delivery is the amount of food delivered to the nest, and the Average Age tells us how old the ants became after 1000 rounds. The simulation terminates when the ant delivers all the food or the round number reaches 1000.

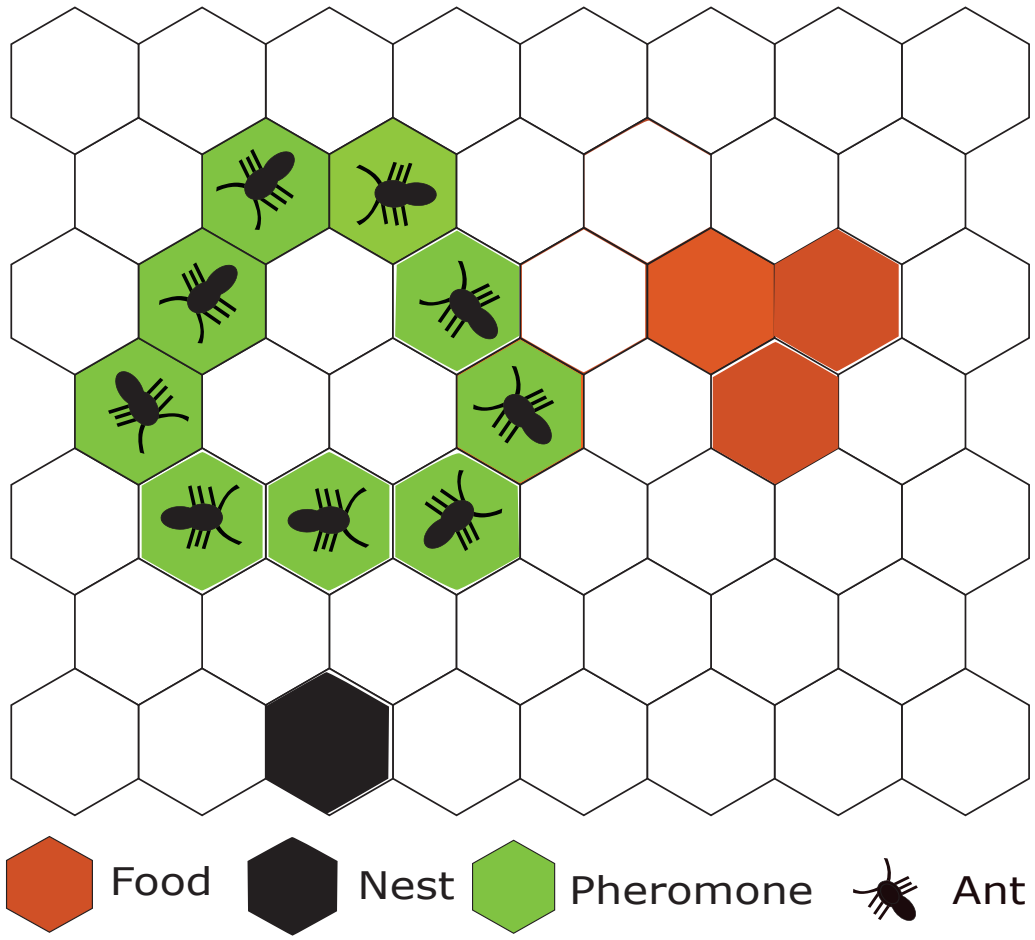


Figure 5.6: Ants got stock into the spiral of death [13]

5.2.3 Contribution

In this paper, we proposed an ant-mill prevention system. This system prevents ants from creating ant-mills. This system avoids ant-mills by scanning the neighbor ants, and if the ants recognize more than two ants, they change their direction to escape. They continue the escape direction for locations and then start the foraging. This paper contributes a unique algorithm for ant foraging to prevent ant-mills. Previous related works on foraging do not focus on the ant-mills. For that reason, the contribution of this paper is the solving of the ant-mill problem; and it proves that this algorithm is more efficient than the standard ant foraging algorithm. Therefore, this paper is an excellent contribution to the science world.

5.2.4 Personal Contribution

Ahmad Reza Cheraghi's contributions to this work include the scientific approach, the first version of the problem definition, i.e., ants foraging, and participating in the solution for the ant-mill. Additionally, he finalized this paper. Under his supervision, Jochen Peters discovered the ant-mill problem and implemented, based on several discussions with Ahmad Reza Cheraghi, the solution for the ant-mill and evaluated it. Further, he wrote the first draft of this paper. Kalman Graffi was continuously involved in discussing the scientific approach of the algorithms and provided a critical revision of the paper.

5.3 Universal 2-Dimensional Arbitrarily Shaped Terrain Marking

This section summarizes the contributions and gives a verbatim copy of our paper [14].

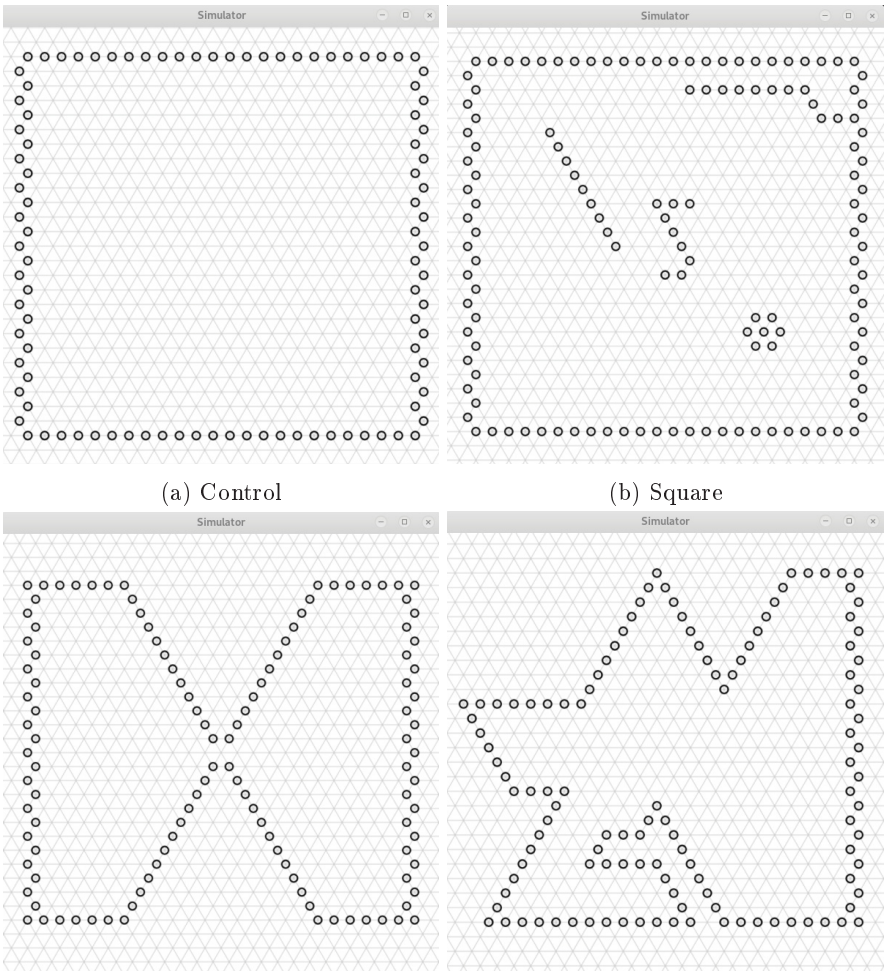
Ahmad Reza Cheraghi, Abdelrahman Abdelgalil, Kalman Graffi
"Universal 2-Dimensional Terrain Marking for Autonomous Robot Swarms".
In: *Proceedings of IEEE 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS))*.
2020.

5.3.1 Paper Summary

This paper discusses how to mark arbitrary terrains with robot swarms. We have designed and implemented two algorithms for solving this problem and evaluated them with the simulator Swarm-Sim. Given are four undiscovered terrains. Each terrain has a different shape and may include obstacles. The robots' aim is to discover and avoid obstacles, and visit each location point in the whole terrain to mark each one of them without missing any. Marking each location helps to tell, indirectly, other robots that a location has been visited, and thus it is not necessary to revisit it. Additionally, the robots can communicate with each other. The evaluation shows that by increasing the number of robots, the efficiency of marking arbitrary terrains increases.

For the marking problem, given are four terrains shown in Fig. 5.7. Each terrain has a different form and a given number of locations. Additionally, they have challenges for the robots to solve. The first terrain is the control terrain. It is a simple square without any obstacles and with 563 location points. Here, the challenge is for the robots to visit and mark each terrain location without any obstacles. The square terrain is the second one with 527 locations and obstacles. For the robots, the challenge here is that they must detect the obstacles and avoid them without marking them. The next terrain is the constricted terrain. It contains two simple terrains, i.e., without any obstacles, connected by a path of the size of one robot. The number of locations in this terrain is 363. The robots' aim is not to enter the path without interfering with each other. The edgy terrain is the last terrain with 363 locations. It is, as the name says, a terrain with many edges. Here, the robots face many corners, and their challenge is to mark all the edges. Now we know the terrains. However, it is essential to talk about the capabilities of each robot.

Before presenting the algorithms, it is necessary to describe the main capabilities of each robot. The first one is that they must move. The robot must move because it must conquer and mark the terrain. Therefore, they should have wheels or legs. Second, the robots should be capable of marking the terrain. The robots need something to mark each location point. This marking object can be a paint marker, chalk, or even a knife, such as cutting grass. Third, a communication tool, such as Wifi or Bluetooth. This tool is necessary for the robots to communicate within a certain radius to exchange the locations of the marked areas, thus avoiding unnecessary multiple markings.



(a) Control (b) Square

(c) Constricted (d) Edgy

Figure 5.7: Terrain Shapes [14].

After defining the robots' capabilities, we now present the algorithms for marking the four terrains. We designed and implemented two algorithms for solving the terrain marking problem: Breadth- and Depth First Marking Algorithm.

Breadth-First-Marking (BFM) is an altered version of the classic Bread-First-Search by adding the First In First Out (FIFO) algorithm for memory reading. First, the robot scans its neighborhood locations and stores all the unmarked locations. Afterward, it visits, based on the FIFO principle, all its adjacent scanned locations. When a robot is in an unmarked location, it will mark it and store its coordinates. Then the robot reads from its memory the following location coordinates and moves towards it. This procedure continues until all the locations in the memory have been marked. After visiting and marking all the scanned locations, the robot receives the coordinates of the first marked location and moves towards it. Then, the same scanning and marking procedure happens again until the robot reaches a wall or nothing

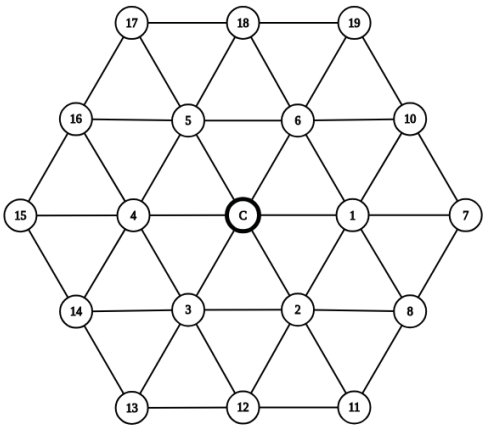


Figure 5.8: BFM

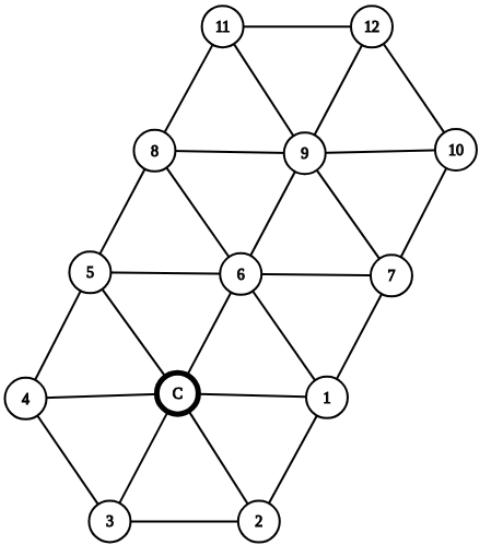


Figure 5.9: DFM [14].

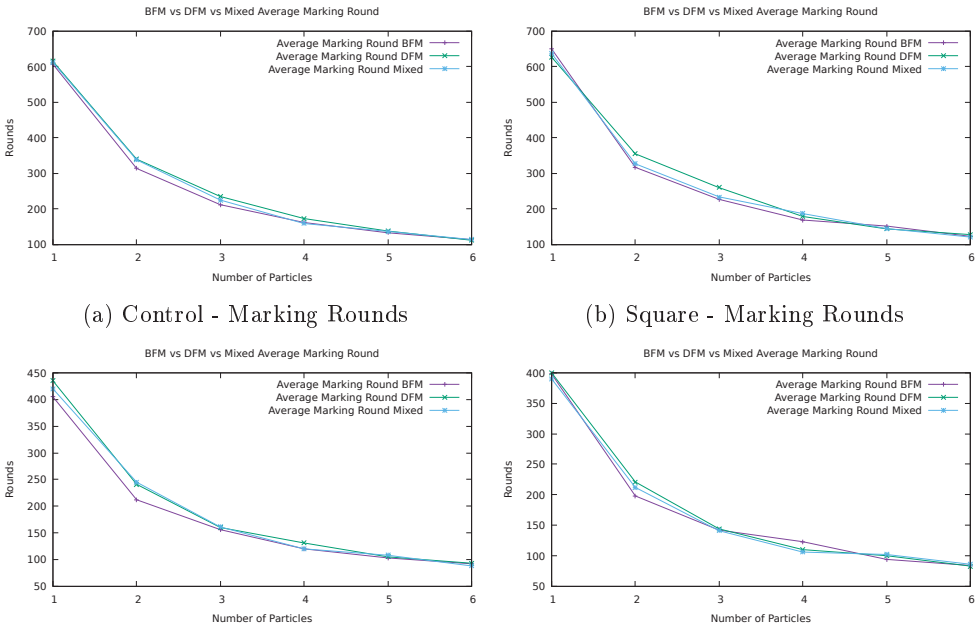
is left for visiting and marking. A finished BFM marking looks like a circle, as shown in Fig. 5.8.

Depth-First-Marking (DFM) is the second algorithm we propose to mark the terrains. The inspiration for this algorithm comes from the Depth-First Search. The DFM algorithm uses a similar mechanism as the BFM. However, in addition, the DFM uses the Last-In-First-Out (LIFO) principle for memory retrieval instead of FIFO. Thus, LIFO causes that the robot to move to the last scanned location instead of the first one, like BFM. Therefore, the robot keeps moving straight in one direction until it reaches any boundary in the terrain. As a result, the DFM of the terrain looks like a straight line, as shown in Fig. 5.9. Now that we described the two algorithms BFM and DFM, we will talk about the obstacle avoidance algorithm.

When a robot faces an obstacle, it stops the marking algorithm and uses the obstacle avoidance algorithm to move around the obstacle. The obstacle avoidance helps the robot move forward without changing the way of marking the terrain. While moving around the obstacles, it still marks unvisited locations until it reaches the other side of the obstacle, then it continues its typical marking algorithm. Now that we know the marking and the obstacles avoidance algorithm, it is time for the evaluation.

For the evaluation, we use Swarm-Sim with the following setup. Given are the four terrains, six robots, and the algorithms BFM, DFM, and mixed. Mixed means that one half of the robot swarm uses the BFM and the other half, DFM. We compare these three algorithms to each other to see if the robot swarm can mark the whole terrain and to check how many rounds they need to accomplish this task.

The metrics for the evaluation are the swarm size and the round numbers. The size of the swarm starts from one and goes up to six. We compare each swarm size to find out how many



(a) Control - Marking Rounds

(b) Square - Marking Rounds

(c) Constricted - Marking Rounds

(d) Edgy - Marking Rounds

Figure 5.10: 1 to 6 Particles - BFM vs DFM vs Mixed [14].

rounds they need to mark a terrain. Before a new round starts, each robot does its scanning, moving, and marking.

We evaluate each terrain with six algorithms, one to six robots, and 20 different seed numbers. Therefore, we have in total, 1440 simulations. The results are shown in Fig. 5.10 and are as follows. Regardless of the swarm sizes, a 100 percent success rate is achieved with all the simulations. Second, all three marking algorithms' performances are almost similar. Only, the BFM is slightly faster with a smaller number of robots. However, it becomes equal to the others as the number of robots increases. The rounds needed by the Mixed are between DFM and BFM. Nevertheless, all of the algorithms become identical to each other as the size of the swarm rises. The only downfall is DFM because its marking strategy is slightly slower. Generally, for all algorithms, the performance increases logarithmically as the count of robots increases. However, there is a tiny difference between the control and square terrains. While the number of robots increases, the rounds for marking decrease by approximately 83 percent. For the constricted and edgy terrains, the rounds need to complete the marking decrease by about 75 percent.

After analyzing the evaluation data, we can conclude that the solutions are robust, as each robot is individually capable of marking the complete terrain; they are flexible, as the algorithms are compatible with all group sizes of the swarm and scalable because of the algorithm's autonomous nature. The solution performs well when the robot group size increases. As for the performance, larger group sizes produced significantly better results. Finally, all three marking algorithms possibilities perform similarly across all simulation setups for each robot group size. However, the BFM performs better with fewer robots, but it becomes comparable

Figure 5.11: Dog marking his territory (Source: [163])

as the number of robots increases. On the other hand, the DFM performs slower with fewer robots. The conclusion is that by having only a few robots, the marking should be done in a circle movement instead of from one direction to another.

For future work, we suggest having real-world scenarios, thus evaluating communication range and frequency limitations. Second, the data between the robots is shared indiscriminately in our solution. An improvement can be that the robots use a filter mechanism for sharing necessary data. This improvement may lead to better task allocation and effort distribution. Another point is to optimize the obstacle avoidance algorithm. Currently, the robot chooses a random direction to rotate around the obstacle, which may cause the robot to take a longer way around. A simple solution to make it an impediment would be to enable the robot to memorize critical points within a terrain such as choking or bottleneck entrances and use them as a reference when encountering obstacles.

5.3.2 Importance and Impact on Dissertation

This paper is the third part of the communication module M4 and covers the direct and indirect communication for robot swarms, which is the contribution C4.3. The inspiration for this algorithm comes from dogs, which is the answer to RQ3.1. Dogs use indirect communication by marking their territory by urinating (Fig. 5.11) and direct communication with barking (Fig. 5.12). The answer to RQ3.2 is that this marking algorithm aims to discover and mark an undiscovered terrain. The following points are the answer to RQ3.3. Given are a swarm of robots and a closed, undiscovered terrain. Robots can move, mark, and communicate with each other. The answer RQ3.4 is that we solve the task with two marking algorithms: Breadth- or Depth-First-Marking. Whenever all its neighbor locations are marked, the robot stops

Figure 5.12: Dog barking (Source: [164])

discovering and marking. Therefore, the marking algorithm is validated when the terrain is completely marked, which is the answer to RQ3.5. The answer to RQ3.6 is that we use the Swarm Sizes and Rounds as the metrics.

5.3.3 Contribution

The contribution of this paper is to introduce two algorithms for robot swarms to mark arbitrary terrains based on Breadth- and Depth-First-Search algorithms. The robots use direct communication for sharing a list of marked location coordinates. Additionally, the robots can communicate directly over Wi-Fi or Bluetooth or indirectly. However, our proposed algorithms let robot swarms communicate indirectly by marking arbitrary terrain: the so-called stigmergy. Stigmergy is a way of indirect communication that animals use. For example, ants use pheromones to indirectly communicate the path to a food source. This paper uses stigmergy to tell other robots with marked locations that they are unnecessary to get marked. Therefore, we provide another way of indirect communication. As a result, our marking algorithms are remarkable and unique because we use direct and indirect interaction to mark arbitrary terrains.

5.3.4 Personal Contribution

Ahmad Reza Cheraghi's contributions to this work include the scientific approach, problem definition, and solution for using marking as indirect communication within the swarm. He also drafted the first version of this work. Under his supervision, Abdelrahman Abdelgalil performed the implementation and evaluation of the marking algorithm. In addition, he authored and completed this work, and he refined the solution in several discussions with Ahmad Reza Cheraghi. Kalman Graffi continuously participated in the discussion of the scientific approach of the algorithms and provided critical revision of the work.

Chapter 6

Dynamic Movements for Robot Swarms

We dedicate the last robot swarm algorithm part to the movement possibilities of robot swarms. The robot swarm is decentralized; rules need to be declared for the robots to move together as a swarm. We first provide the three-zone model. This model allows the robots to stay within the swarm. Next, we present the phototactic movement. The swarm robots move whenever they sense light. However, they do not know from which direction the light emission is coming.

6.1 Robot Swarm Flocking on a 2D Triangular Graph

Ahmad Reza Cheragh, Asma Ben Janete, Kalman Graffi
"Robot Swarm Flocking on a 2D Triangular Graph".
In: *Proceedings of IEEE 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS))*.
2020.

6.1.1 Paper Summary

In winter, the birds fly together as a flock from north to south. They stay close together without interfering with each other. Flocking helps them look out for each other and gain momentum as they fly. However, it is interesting to know how this flocking behavior works and how to develop and simulate such a phenomenon. In this paper, we present flocking for robot swarms in a hexagonal field.

Given is a swarm of robots that are close to each other like a herd with limited ability. They have no GPS, and therefore, do not know where they are. Communication between the individual robots is only possible within a specific range. All robots are homogeneous and, therefore, have no leaders or central units. The robots have sensors with which they can check their surroundings and calculate their distances from each other. This work aims to simulate the flocking behavior of robot swarms and solve the flocking problem. The flocking problem defines a group of agents that move together as a swarm without interfering with each other or separating, and maintaining a constant velocity. To solve this problem, we use an adapted three-zone swarm model of Couzin et al. [165].

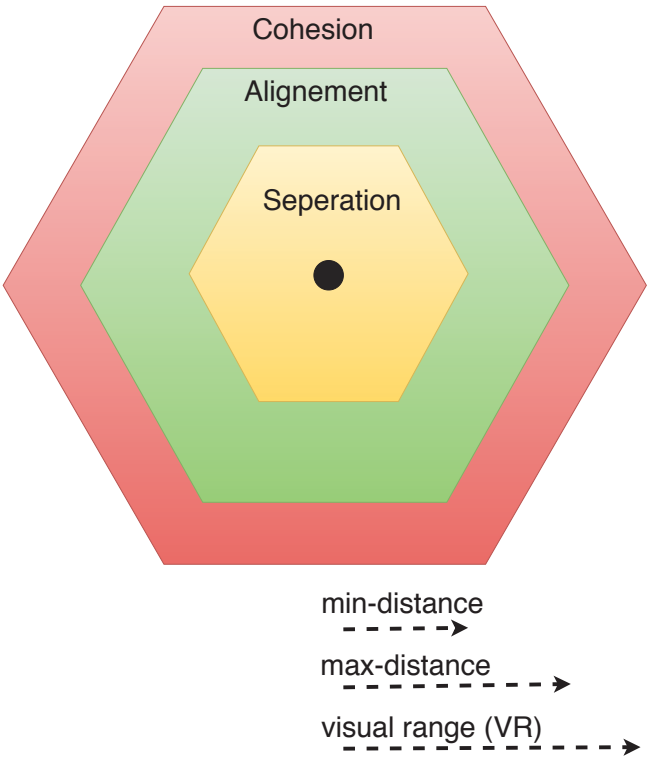
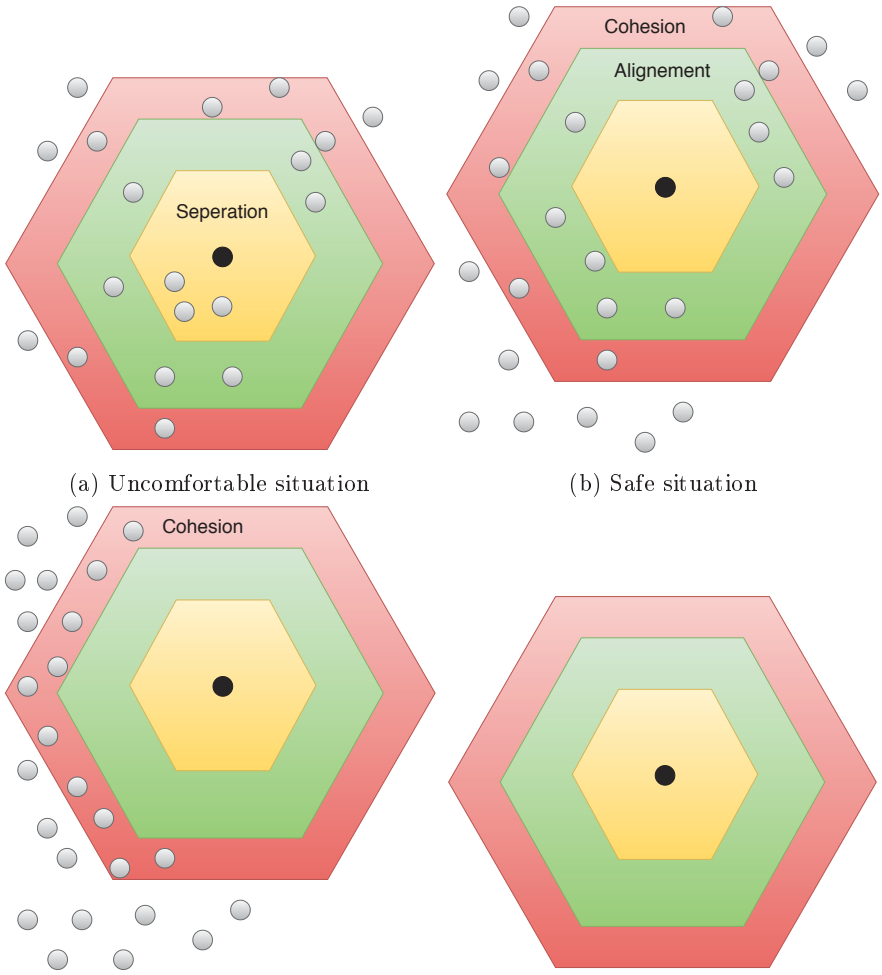


Figure 6.1: A particle with its 3 interaction rings [15].

The adapted three-zone flocking rule model is shown in Fig. 6.1 and consists of three zones. The first is the yellow zone, which is called separation. Neighbor robots that are in this zone are too close. Therefore, the robot that recognizes that its neighbors are in this zone must wait to let them out of this zone. The second zone, the alignment zone, is the ideal zone (it is given a green color). Each robot must make sure that its neighbors are in this zone. The last zone is the cohesion zone. Here, the robots are on alert, so the zone is colored red. The robot is in danger of being lost from the swarm; therefore, it must move faster and towards the direction of the swarm to not get lost from the flock. However, we have defined a set of distances that make up the three zones for each robot.

The first distance is the "min-distance," which defines the minimum distance between one robot and another. The "max-distance" defines, on the other hand, the maximum distance that should be between each robot. The third range is the visibility range, which specifies the distance for scanning neighborhood robots. The separation zone is defined by the distance from a robot to "min-distance." The distance between the min-distance and max-distance is the alignment zone. Finally, the cohesion zone is the distance between the max-distance and the visual range (VR). The unit for the width is "hops." A hop is the distance between a location and its adjacent location.

Four situations arise for a robot with these three zones and distances. When the robots are too close, they are inside the separation ring, and the robot is in an uncomfortable situation



(a) Uncomfortable situation (b) Safe situation

(c) Critical situation (d) A lonely particle

Figure 6.2: Possible situations of a particle in the flock [15].

(Fig. 6.2a). The ideal situation is the safe situation (Fig. 6.2b)- In this situation, most of the robots are in the alignment zone, regardless of how many are in the cohesion zone. The critical situation is alarming (Fig. 6.2c). In this situation, all the robots are in the cohesion zone, which means in an alarm state, so they need to catch the flock. Otherwise, they become lonely. The lonely situation is the robot's worse situation (Fig. 6.2d) because no robots are in one of the rings. Therefore, it is not in the swarm anymore. Based on these three zones, the robot decides how to behave and move.

The three-zone flocking algorithm works as follows. First, the robot scans its environment for robots in its field of view and lists the robots it finds along with their distances. Next, it uses the information gathered to check its current situation, i.e., lonely, uncomfortable, safe, or critical, and decides what to do next. If it is in a lonely situation, it moves randomly in any direction. Otherwise, when a robot is uncomfortable, it either stays with no space or checks

where the minority are and follows them. In a safe situation, the robot balances movement and velocity. It checks in which direction the majority is and follows them. Finally, in a critical situation, it will choose to move faster to keep itself in the herd and avoid becoming lonely. In this case, the robot communicates with its neighbors to determine which direction they are going. Then it follows the direction where the majority is moving. We developed and simulated this flocking algorithm.

For simulating the three-zone flocking algorithm, we use the Swarm-Sim simulator with a triangular graph. Each point on the triangular graph is called a location, and the distance between adjacent locations is a hop. The swarm has a size of 105 robots. We use different widths for the separation, alignment, and cohesion rings to determine if these widths result in the robot staying within the flock for a longer time. Therefore, the evaluation is stopped whenever one of the robots enters a solitary situation or after 1,000 rounds. Additionally, the flocking algorithm uses breadth-first search to check if all the robots are in the swarm and terminates if one is missing.

The metric for the evaluation is success rate. The formula for the success rate is the number of rounds divided by the maximum round number. The unit is percentages. Thus, it tells how long, in a percentage, the flock was stable and not separated within the 1,000 rounds.

As we mentioned earlier, the evaluation setup is based on the different widths of the three zones. The robots start as a flock. Then all of them move and stay in the flock, i.e., within the three zones. We first start the changes in the width of the separation ring. The same is done with the alignment and cohesion ring. As a result, we have three experiments to show how the changes of the widths affects the flocking behavior of the robot swarm.

The first experiment is the impact of the separation ring. The parameters of the first experiment are shown in Table 6.1. After each simulation, the min-distance is increased by one hop. The max-distance and virtual range are changed respectively to keep the width of alignment and cohesion ring constant. We start the simulation with a width of zero jumps and stop it after the sixth width. Table 6.4 shows the success rate results. For any width between zero and five jumps, the success rate is 100 percent. However, with a width of six, the success rate decreases to 98.9 pecent. It is an exciting phenomenon. The large separation ring lets more robots into it, which causes them to get separated more easily. However, each of them follows the minority direction but only if there is a space in that direction. Hence, there are now more inside the separation ring. Those in the alignment or cohesion ring cannot move because a robot in the separation ring still occupies the space in that direction. Consequently, they must stay and become lonely, which causes an early termination.

In Table 6.2 the setup of the second experiment is shown. Here the width of the alignment ring is increased. The min-distance stays constant. Max-distance and VR are increased each by one and up to six. The result of this experiment is a 100 percent success. As a result, the alignment ring has no impact on the cohesion flocking. The last experiment is the changing of the cohesion ring width. Only VR changes and min-distance and max-distance stay constant as shown in Table 6.3. The result of the success rate is shown in Table 6.3. Here the success rate starts small and increases with the width change. At a width of zero, the success rate is 14.9 percent. It shows that when there is no cohesion ring, the flock is separated quickly. However, with a width of one, the success rate is 42 percent. The success rate becomes 99.7 percent with a width of two. From a width of three onwards, the success rate becomes 100

Table 6.1: Parameters of 1^{st} experiment [15].

No	min_ dist.	max_ dist.	VR	Width of the Separation Ring
1	1	3	7	0 (No separation)
2	2	4	8	1
3	3	5	9	2
4	4	6	10	3
5	5	7	11	4
6	6	8	12	5

Table 6.2: Parameters of 2^{nd} experiment [15].

No	min_ dist.	max_ dist.	VR	Width of the Alignment Ring
1	2	2	6	0 (No alignment)
2	2	3	7	1
3	2	4	8	2
4	2	5	9	3
5	2	6	10	4
6	2	7	11	5

Table 6.3: Parameters of 3^{rd} experiment [15].

No	min_ dist.	max_ dist.	VR	Width of the Cohesion Ring
1	2	4	4	0 (No cohesion)
2	2	4	5	1
3	2	4	6	2
4	2	4	7	3
5	2	4	8	4
6	2	4	9	5

percent. To sum up, the success rate is low if there is no cohesion ring, and it increases as the width increases.

The conclusions of the experiments are, first, that increasing the width of the separation ring causes the flock to become separated. Second, an increase in the alignment ring width causes no harm to the flock. Last, a small cohesion ring causes the robots to separate faster. Only with a width of three and above does the success rate stays at 100 percent.

The simulation of the flocking only occurred in a 2D triangular field. However, the real-world is not a plain and triangular graph. Thus, it is interesting for future work to adapt this flocking mode in a 3D world. Another future work could be to upgrade the flocking algorithm for detecting obstacles or preventing predators from being separated and coming back together. Nevertheless, we present a three-zone flocking algorithm for robots in a triangular field. We experimented with different widths in the three-zone and their effects on the flock and concluded that changes in the width would affect the flocking behavior of a robot swarm.

Table 6.4: Success rate for different widths of separation ring [15].

Width of separation ring	0	1	2	3	4	5
Success rate	100 %	100 %	100%	100%	100%	98,9%

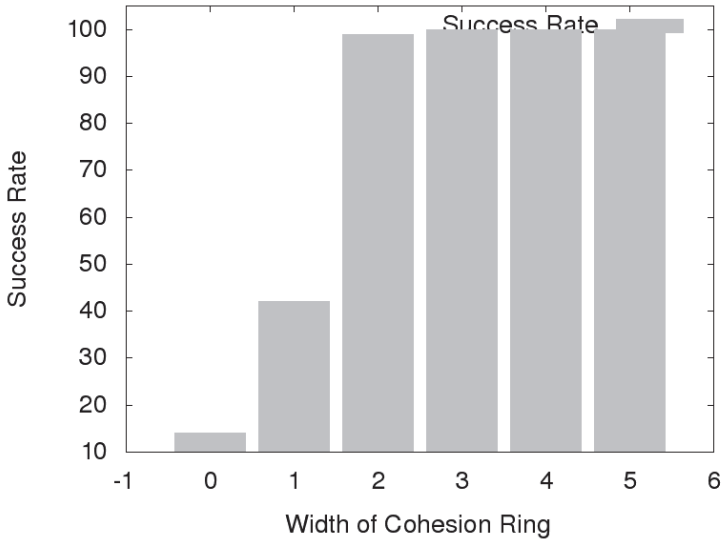


Figure 6.3: Success rate with different cohesion ring widths [15].

6.1.2 Importance and Impact on Dissertation

Module M5 is about the movement of robot swarms, and this paper contributes rules for the robot swarm C5.1. We present the movement of many entities inspirit from birds flying together as a flock (Fig. 6.4) and as mentioned earlier in chapter 1.1 from fish shoal (Fig. 1.3), which answers RQ3.1. The. It is fascinating to see how they do not interfere with each other, stay together, and change direction together. To answer RQ3.2, the aim of this paper is to define rules that lead robots to be within their swarm while moving. Given are robots limited in their vision, sensing, and communication ability, which is the answer to RQ3.3. To answer RQ3.4, we use a three-zone model to reach the aim. This model defines three zones based on the distance to the robot's neighbor. The robot keeps its speed, speeds up, or slows down based on which zone it currently moves. We test flocking with a variation of the zone width and the robot's vision to find out which one of the combinations is optimal. To answer RQ3.5, flocking is validated when the robots are still within the flock after 100 rounds. The metrics we use are the Swarm Sizes and the Rounds, which is the answer to RQ3.6.

Figure 6.4: Bird flock (Source: [166])

6.1.3 Contribution

The flocking phenomenon is well known in the science world. Additionally, birds fly together as a flock from north to south in the winter time without interfering. The contribution of these papers to the science world is that this flocking algorithm uses an adapted three-zone model on a triangular field. Secondly, it experiments with how different widths can affect the flocking behavior. These experiments with a triangular grid and different widths are new and an excellent contribution to science.

6.1.4 Personal Contribution

Ahmad Reza Cheraghi's contributions to this work include the scientific approach, problem definition, and solution of the flocking on a 2D triangular grid. Additionally, he wrote and finalized this paper. Under his supervision, Asma Ben Janete did the implementation and evaluation of the flocking algorithm. In addition, the solution was refined in several discussions with Ahmad Reza Cheraghi and Asma Ben Janete. Kalman Graffi was continuously involved in discussing the scientific approach of the algorithms and provided a critical revision of the paper.

6.2 Phototactic Movement of Battery-Powered and Self-Charging Robot Swarms

This section summarizes the contributions and gives a verbatim copy of our paper [16].

Ahmad Reza Cheraghi, Fabio Schloesser Vila, Kalman Graffi
"Phototactic Movement of Battery-Powered and Self-Charging Robot Swarms".
In:*Proceedings of IEEE 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS))*.
2020.

6.2.1 Paper Summary

Phototaxis is a biological mechanism in that light influences organisms' movement directions, even though the light emission direction is unknown. One example of such an organism is the trochophore larvae (zooplankton). They have a unique sensor for detecting light without seeing. The jellyfish is another example. Their sensors prevent them from predators when they feel a shadow.

This paper aims to imitate phototactic behavior on two types of robot swarms: battery-powered and self-charging. Battery-powered robots move when they sense light and self-charging move after they are charged from the light. The rule for the phototactic imitation is that they are only allowed to move when they feel light, and those in a shadow position -overshadowed by ones that felt light- are not allowed to move. Each robot swarm starts in the middle of a hexagon field with a finish line and light emitters coming from one direction. We aim to pit the self-charging against the battery-powered robot swarm to see who reaches the finish line within 5,000 rounds.

An example of our problem statement is shown in Fig. 6.5. The robots are positioned like a hexagon in a hexagon field. The light is coming from the left side, and the aim is to bring the robots to pass the finish line (blue circles) on the right, as shown in the second picture of Fig. 6.5.

We have two types of robot swarm. The first type is battery-powered robots, and the second is self-charging robots. Battery-powered robots move, without waiting whenever they sense light. The robots that come out from the shadow of the neighboring nodes on their left will move too; this is different from the self-charging robots. The photovoltaic/solar cells-equipped, self-charging robots, need to be charged before they can move. Thus, they must stay for some time before moving whenever they receive light. The battery-powered robots, on the other hand, immediately move when exposed to light. However, these two robot types have the following common attributes:

1. All are using the same compression algorithm, i.e., staying constantly connected within the swarm.

2. They cannot communicate with each other.

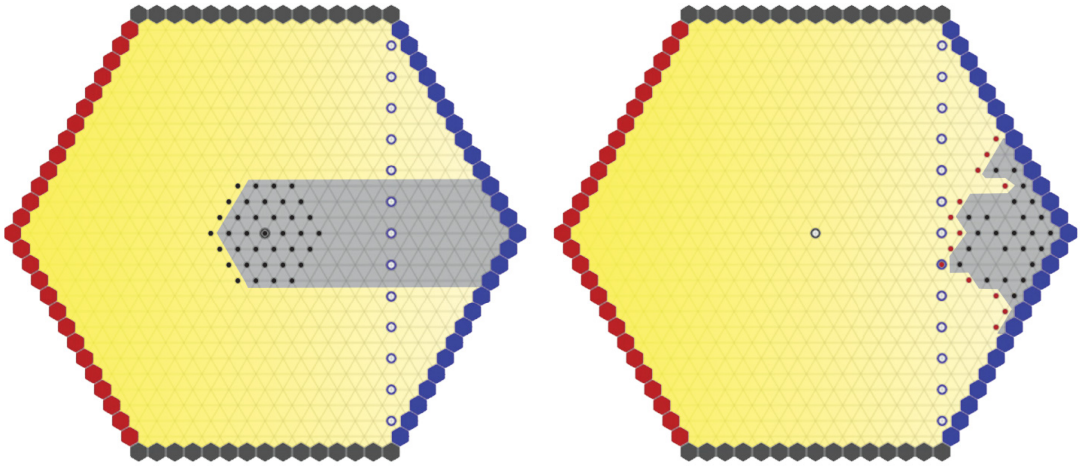


Figure 6.5: Example for a swarm phototaxing simulation [16].

3. Through a binary sensor, the light is felt.

4. They can feel the nearby robots.

5. Before each movement, each robot must check the connectivity within the swarm.

6. A movement occurs whenever they receive light.

7. The direction of the movement is random.

Knowing the attributes and the details of those types of robot swarms, we aim to challenge them to see who will reach the finish line within 5,000 rounds. The evaluation aims to compare the battery-powered with the self-charging robot swarm with different swarm sizes. The robot swarm size varies between two to 30 robots, and for each swarm size, we simulate ten times with a different seed number. Seed numbers are essential for changing the randomness of direction within the simulator. The simulation terminates after 5,000 rounds or when the swarm passes the finish line.

We simulate this competition with the Swarm-Sim. The starting scenario (Fig. 6.6) is a hexagon field with a finish line (blue marked circles) on the right-hand side with light coming from the left. The robots are the black dots in the middle of the field. The shape of the swarm is a hexagon. Thus, we have robots receiving immediate light, as well as robots in a shadow position (not receiving immediate light) because of the robots near the light. Robots in a shadow position will not move. They only will move when they are in a light position and sense light. Nevertheless, it is vital to know how to measure the competition. For measuring the competition, we use the following metrics. Each metric is the average calculated from the ten simulations of each swarm size. Additionally, for each metric, we generate the "Standard Deviation" (SD) to express how far the error for each measured metrics is from its mean.

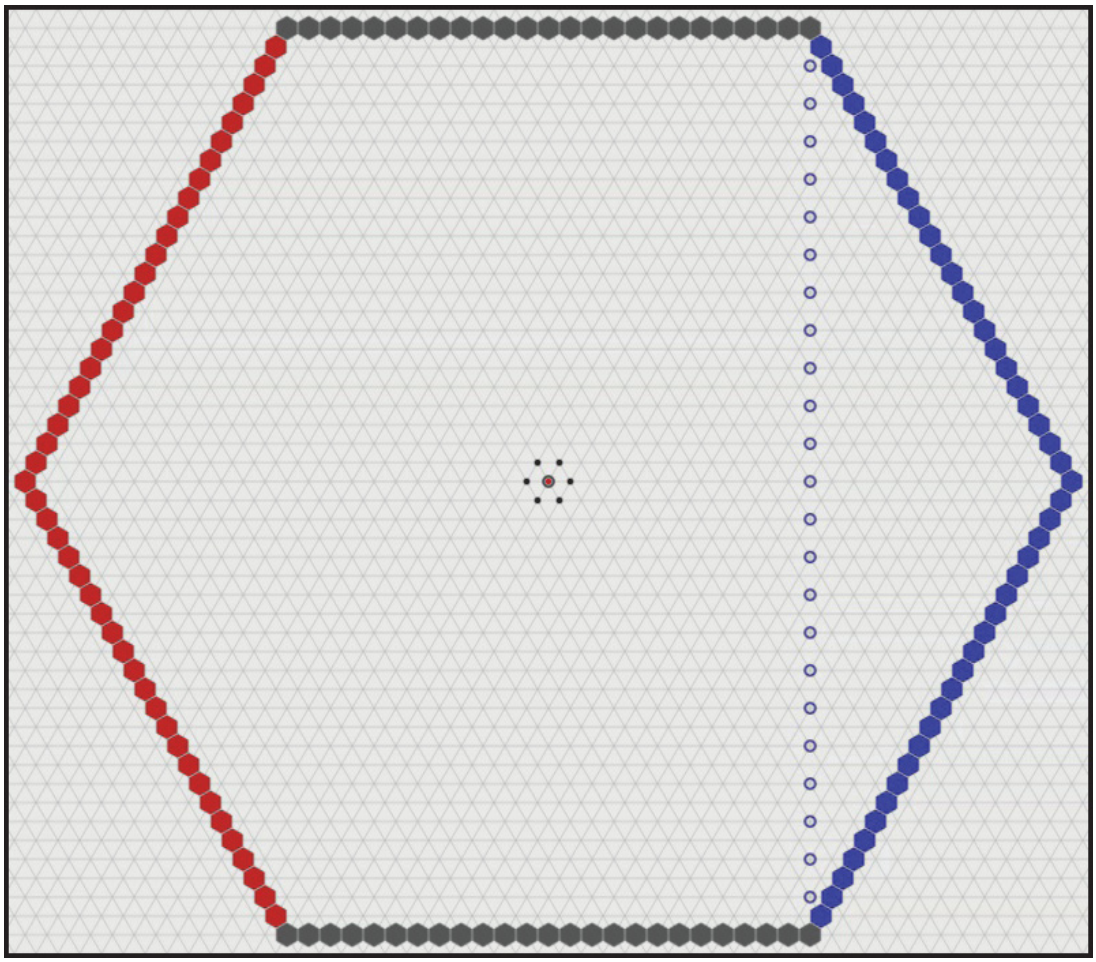
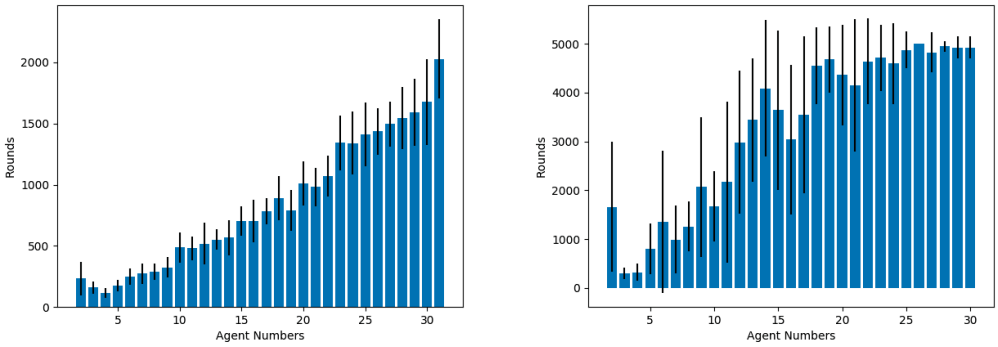


Figure 6.6: Test environment

Nevertheless, it is vital to know how to measure the competition. For measuring the competition, we use the following metrics. Each metric is the average calculated from the ten simulations of each swarm size. Additionally, for each metric, we generate the "Standard Deviation" (SD) to express how far the error for each measured metrics is from its mean. The first metric is the "Success Rate," which tells us in percentage the average of successful passing of the swarm over the finish line. This metric varies between 100 percent and 0 percent. When the swarm crosses the finish line in all ten simulations, the success rate is 100 percent.

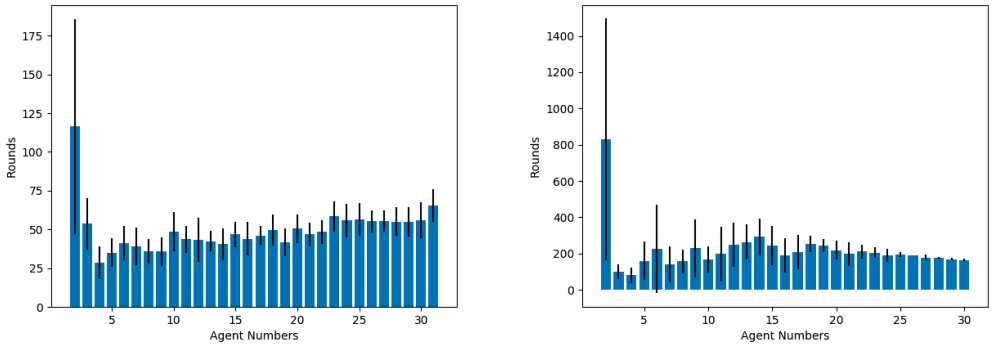
We use the metric "Average Rounds"(AR). It tells the rounds it took for the robot swarm to cross the finish line for ten rounds. However, if the AR is 5,000 rounds, it means that the swarm never reached the finish line over the entire ten simulations. Everything below 5,000 rounds means that the swarm passed the finish line at least once.

Moreover, the "Average Rounds Per Agent" (ARPA) is identical to AR, with the difference that it shows us the time for each robot. ARPA helps us understand if the swarm size affects the time it takes each robot to reach the finish line.



(a) Self-charging agents (b) Battery-powered agents

Figure 6.7: Average rounds with standard deviation [16].



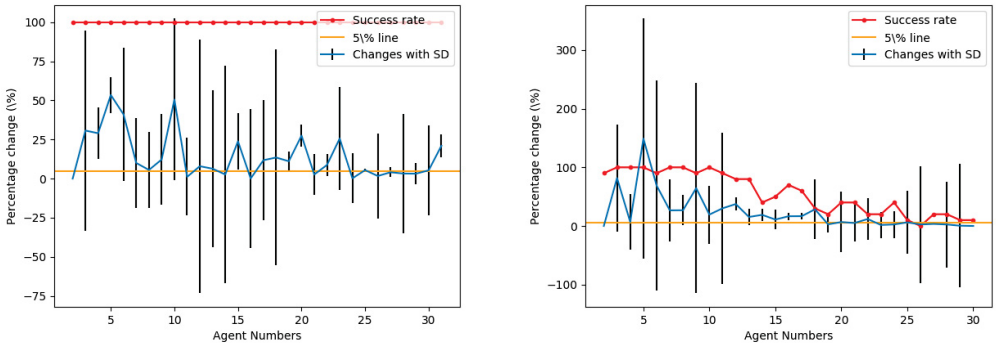
(a) Self-charging agents (b) Battery-powered agents

Figure 6.8: Average rounds per agent with standard deviation [16].

It is interesting to see how much the time varies between each swarm size. For this purpose, we use the "Percentage Change Average Rounds" (PCAR) and "Percentage Change Average Rounds Per Agent" (PCARPA) metrics to identify the percentage change of time between each different swarm and for each agent. We assume that the greater the swarm size, the more time changes become identical.

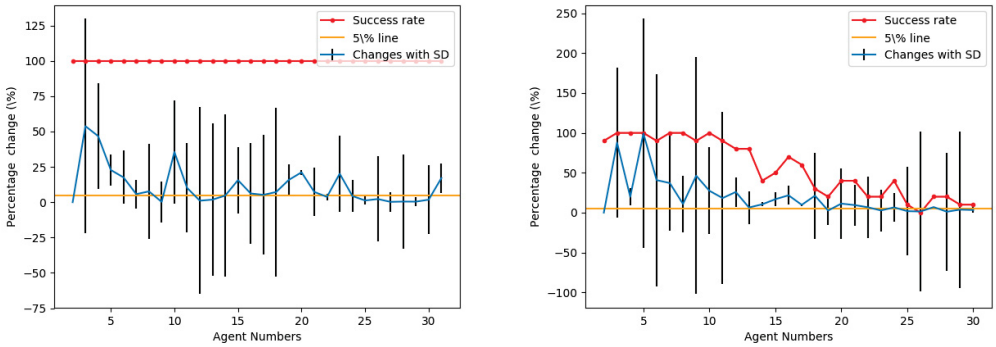
After understanding the metrics, let us observe the simulation results from Figures 6.7, 6.8, 6.9, and 6.10. The bar charts in Fig. 6.7, 6.8 show each swarm size and its type, the AR and ARPA. The changes, PCAR & PCARPA, between each swarm size and for each type are shown on the graphs in Fig. 6.9, and 6.10. We elaborate on the time each swarm size and type needed on average and the changes between each swarm size.

The first bar chart Fig. 6.7a is for the self-charging swarm. The x-axis represents the swarm size, and the y-axis– the time AR. This bar chart shows how much time each swarm size needs to reach the finish line. In the beginning, with a swarm size of two, the time is much longer



(a) Self-charging agents (b) Battery-powered agents

Figure 6.9: Percentage change average rounds with standard deviation [16].



(a) Self-charging agents (b) Battery-powered agents

Figure 6.10: Percentage change average rounds per agent with standard deviation [16].

than the following ones. Afterward, the time drops by three and four robots, and then it starts to increase as the swarm size increases. The high AR with a swarm size of two is normal because, at this size, the robots nearly do not produce shadows for each other, and thus the two robots continuously move around each other instead of moving forward towards the finish line. We have a sudden decrease of time by the swarm size of 19 and 21 because the swarm is moving randomly. Thus, they are seldom receiving better directions based on the bias of the swarm simulator. But hence looking at the SD shows that it is more than the previous one, and thus, the time becomes longer as the swarm size increases. In general, the self-charging robots' highest time is almost 2,000 rounds. In conclusion, the self-charging swarm is always successful, as passing the finish line is below 5,000 rounds, and the time increases with the size of the swarm.

The second bar graph for the battery-powered robot swarm (Fig. 6.7b) looks like the self-charging robot swarm at first glance since the AR increases with the size of the swarm. However, comparing the time with the self-charging robots, the battery-powered robot swarm size takes

at least twice and at most ten times as long as the self-charging. All, except swarm size 26, of the swarm size could cross the finish line at least once as they are under 5,000 laps. The time for swarm size 26 is 5,000 rounds, which means that within the ten simulations, none of the 26 robot swarms could cross the finish line.

The result in Fig 6.9, 6.10 presents the changes of time in percentages between each swarm size. The x-axis is the swarm size, and the y-axis is the percentage change. The red line in the graph is the success rate. The success- rate for all self-charging swarm sizes is 100 percent. Second, as the swarm size increases, the PCAR approaches the 5 percent line. As a result, it shows the more robots added to a swarm, the better they move away from the light. The success rate of the battery-powered robot swarm decreases as the swarm size increases. However, the PCARs also tends to be 5 percent and more minor than the success rate, i.e., reaching the finish line within 5,000 laps, decreases with larger swarm size, thus equalizing the changes among them.

For each robot, the results are shown in Fig. 6.8a, 6.8b. The first bar chart in Fig 6.8a shows the ARPA. For the self-charging robots, the ARPA increases slightly as the number of robots increases. However, for the battery-powered robots (Fig. 6.8b), the time rises to a point where the success rate becomes low. Then ARPA becomes almost stable, as most of the following swarm sizes are not able to reach the finish line within 5,000 rounds over the ten simulations.

The PCARPA in Fig. 6.10 for both battery-powered and self-charging swarms decreases as the number of robots increases. Further, the PCARPA lands far below the 5 percent line. However, this result for the battery-powered is because of the limitation of 5,000 rounds.

The results we present show that the self-charging swarm is more than twice faster than the battery-powered swarm for reaching the finish line. The reason is that the robots in the self-charging swarm do not interfere with each other. While with the battery-powered robots, they move whenever they sense light, which causes chaos, thus the time for reaching the finish line increases. With both swarm types, the time increases as the number of robots increases. The reason is that the complexity within the swarm increases. However, observing the time for each robot shows that the changes are small between each swarm, especially for larger swarm sizes.

Some future work can be done as follows. One limitation in this evaluation is the robots' positions because we arranged the robots as a hexagon. Thus, we can set them differently to compare how these different positions will affect the time for reaching the finish line. Next, work can be done to increase the efficiency, i.e., eliminating redundant steps, e.g., memorizing the previous location. Third, it would be beneficial to upgrade the agents with obstacle avoidance and see how they act on terrains with obstacles. Last, future work to simulate the algorithm in a 3D environment would be a logical next step. These are the future works of this paper, and the conclusion is as follows.

This article concludes that self-charging robot swarms tend to mimic phototactic behavior in nature. Even though self-charging robots only have a binary light sensor, meaning they do not know where the light is coming from, they could move away from the light and reach the target in a time fewer than 5,000 rounds. The results show that as the number of robots increases, the time increases, but when comparing the swarm sizes, the change in time is almost the same; Because as the number of robots increases, they interfere with each other less.

Figure 6.11: Moths seeking for light (Source: [167])

On the other hand, the battery-powered swarms become much slower as the number of robots increases. The robot swarm reaches the finish line later than the self-charging swarm because their sudden movement causes the battery-powered robots to interfere with each other. Thus, this brings chaos, and therefore, the battery-powered robot swarm is slower than the self-charging one. We can avoid this chaos by giving them a timer to not moving immediately after receiving light. However, using a timer consumes power, too. To sum up, the phototactic algorithm is most efficient if the robots are self-charging because they do not cause chaos by not moving immediately, compared to the battery-powered swarm.

6.2.2 Importance and Impact on Dissertation

With this paper, we present for module M5, the contribution C5.2. This paper elaborates on the movement of robot swarms based on light influences (Phototaxis). To answer the RQ3.1, insects such as moths that seek the light in the dark (Fig. 6.11) are the nature-inspiration for this algorithm. Nevertheless, this paper simulates phototactic behavior on battery-powered and self-charging robot swarms. The robots do not know from which direction the light is coming, and the aim is that they should move away from the light and reach the finish line, which is the answer to RQ3.2. The answer to RQ3.3 is robots with the capability of moving and sensing light. The robots have an algorithm to check if they are connected within the swarm. The robot swarm is positioned in an environment with a light bulb that emits light rays and a finish line. The finish line is located opposite the light bulb. We answer question RQ3.4 as follows. With the battery-powered swarm, the robots will sense light and immediately move. With the self-charging robots, they will move after being charged. The answer to RQ3.5 is when the swarm crosses the finish line, this task algorithm is validated, and the aim is reached. We test this phototactic algorithm with different swarm sizes and compare them. For the evaluation, the simulators terminate when the maximum round number, i.e., 5,000 rounds, has

been reached or when the swarm crosses the finish line. The following metrics are dedicated to answer RQ3.6. The Success Rate gives the average percentage of successful passing of the finish line over ten simulations. The metric Average Rounds is the average time of rounds needed for ten simulations. Next, the Percentage Change Average Rounds (PCAR) shows the difference, in time, of two different swarm sizes. The Average metric Rounds Per Agent (ARPA) presents the average round needed for each robot. The last metric is the percentage Change Average Rounds Per Agent (PCARPA), which shows the changes in time between two robot swarm sizes.

6.2.3 Contribution

The paper "Phototactic Movement of Battery-Powered and Self-Charging Robot Swarms" contributes an excellent example of how phototactic movement can work in an environment with robot swarms by comparing the battery-powered robot swarms with self-charging robot swarms. There are only a few related works; however, they all use battery-powered robots compared to our work. In our work, we use both battery-powered robots and self-charging robots. These robots charge after receiving light. Thus, they need some time until they move. As a result, the evaluation shows that self-charging robot swarms have a success rate of 100 percent compared to battery-powered robots. Thus, this paper's contribution is vital for phototactic behavior among robot swarms because it shows that by taking some time before moving, the success rate of phototactic robot swarms will be 100 percent.

6.2.4 Personal Contribution

The contributions of Ahmad Reza Cheraghi include the scientific approach, problem definition, and solution. He also tested and evaluated the phototactic algorithm and authored and completed this work. Under his supervision, Fabio Schloesser Vila implemented the code and participated in the writing of this paper. In addition, the solution was refined in several discussions with Ahmad Reza Cheraghi and Fabio Schloesser Vila. Kalman Graffi was constantly involved in discussing the scientific approach, the algorithms, and the work.

Chapter 7

Conclusion and Future Work

In this chapter, the contents of this dissertation are summarized, followed by a general outlook on the possible future research work. We then present a few closing words to conclude the dissertation.

7.1 Conclusion

In this dissertation, three modules for robot swarms are presented. The first module M1 is the literature survey of robot swarms. Our contribution C1.1 is a survey [7] based on 217 papers. The aim for module M2 is first to analyze the actual state of simulation tools, and second the implementation of the simulator Swarm-Sim. The contribution C2.1 gives an overview of simulators [8] for the mobile ad-hoc, peer-to-peer and opportunistic networks; and the second contribution C2.2 is about the simulator Swarm-Sim, including its mathematical model. For the Swarm-Sim, we published one paper [9]. Inspired by nature we extract three modules. Module M3 is about robot swarm coating objects which gives us two contributions, i.e. C3.1 and C3.2. For this module, we published two papers [10, 11]. Communication within the swarm is handled in the fourth module M4 and includes three contributions (C4.1, C4.2, and C4.3) and publications [12, 13, 14]. Finally, in module M5 we contribute two algorithms (C5.1 and C5.2 for robot swarm movements including two published papers [15, 16].

A brief overview of the past, present, and future of swarm robotics is presented in chapter 2. Over 217 papers are read through and given a summary of swarm robotics' past, present, and future. The pioneers and main definitions of robot swarms are considered. We present various of applications fields, the actual state of real robot swarms, and simulators. Finally, we give a vision of the future of robot swarms. Based on this survey, it is clear that the robot swarm is a research area that is still not fully completed, and some research is needed, especially in swarm simulators and algorithms.

Once the theory of robot swarms is known, we researched network simulators to determine the difference in simulators and what essentials are needed to build a new simulator (Chapter 3.1). We discovered that most of the simulators are networks simulators, most of them are not updated, and many are written in C++, C, or Java. Therefore, we decided to develop a new simulator. We built the robot swarm simulator Swarm-Sim, and the summary can be read in chapter 3.2. Swarm-Sim is a simulator that is easy to learn and program. It has only three

matters: Locations, objects, and subjects. Locations are the points on which either a subject or object can be. An object cannot act on its own. Thus, it is passive and can be used as a ground to walk on; It can be interacted with and picked up. On the other hand, the subject is the leading actor who can perform actions, such as walking, taking things, or communicating. In summary, Swarm-Sim allows these three matters; and a few lines of Python code is used to simulate robot swarms. It is open-source, and thus simple to extend with new grids or different matters; one can also add more features. With Swarm-Sim, we developed, evaluated, and tested more than twelve different algorithms for robot swarms. However, we published seven algorithms for three various application tasks.

We categorized three application fields for the nature-inspired robot swarm algorithms. The first application field is the coating of objects (Chapter 4). Here the robot swarm needs to coat a given shaped object. The aim is that all the robots coat an object as tightly as possible. The coating is successful when the maximum distance from the object of a robot within the swarm is smaller (or equal) than the minimum distance of all accessible locations. For this application field, we programmed two coating algorithms. The first coating algorithm is the leader-based algorithm. This algorithm uses a leader within the swarm. The leader first scans the object, calculates the number of needed robots, then takes each robot from the swarm to coat the object. This algorithm faced four challenges in the completion of coating cave-shaped objects. We show a step-by-step guide on how to solve those challenges. We provided a coating algorithm that works for both cave-shaped and simple objects. However, this is the limitation of this algorithm because it cannot coat any arbitrarily shaped object. Therefore, we developed a second coating algorithm that can do this. The general coating algorithm uses the entire swarm instead of just one leader. The algorithm starts whenever the swarm hits an object, and then each robot moves towards that object to coat it. The closest robot to the object makes space for the robots that are further away. This procedure continues until all the robots are coated as closely as possible. Therefore, the distance of each robot must be defined. The uniqueness of this algorithm is that it works without any distance measurement. The robots calculate their distance. Each robot is initialized with an infinite length. However, if it hits an object, it is given the distance one. Then it starts to share its distance with its neighbor. With this simple technique, the robots can calculate their distance. We tested the general coating algorithm with different swarm sizes and arbitrary object formations. We proved that this algorithm works with any arbitrarily shaped object and added that it is robust, scalable, and flexible. The use-cases for robot swarm coating could be, for example, injecting nanobots into the body to cure diseases such as cancer or destroying viruses. Nevertheless, communication between the swarm is necessary, too.

The second application task is dedicated to swarm communication (Chapter 5). We introduced three algorithms for direct, indirect, and combined communication. First, we used direct communication with the help of opportunistic networks (OppNet). We adapt two OppNet routing protocols: Epidemic and ProPHET, and compared them to each other. The result shows that Epidemic routing is superior in delivery ratio. However, this is based on the cost of low efficiency and high overhead. In contrast, PRoPHET has a lower delivery ratio but is highly efficient and has a lower overhead. Next, we developed an algorithm for simulating ant foraging with the protection of avoiding ant mills. Here the ant uses pheromones for communication. The pheromones tell other ants that a food source has been found and that they should follow the pheromone path. We analyzed food foraging with and without pheromones. We conclude that foraging is less efficient with pheromones because a phenomenon termed the ant-mill occurs. Sometimes pheromones interfere, causing the ants to move in a circle. This circle is

either called an ant-mill or the spiral of death. With this phenomenon, the ants continuously walk within the circle in search food; when they do not find food, they eventually die of starvation. However, we developed an ant-mill protection system. Whenever an ant sees that they are more than two ants in front of it, it runs away from them by changing its direction. Based on this algorithm, we could prevent the ant mill increasing the ants' life duration and make foraging the most efficient. These three types of communication are new for the robot swarm. The last algorithm combines direct and indirect communication to mark arbitrary terrains. The swarm aims to mark the landscape efficiently and avoid obstacles. The swarm uses two marking algorithms: breath-first-marking (BFM) and depth-first-marking (DFM). This marking algorithms differ in marking—the robots using DFM walk in one direction until they hit an end and then use another path. With the BFM, the robots move in a circle. With these marking algorithms, the robots indirectly tell other robots which landscape has already been investigated. Additionally, all the robots are equipped with wireless communication tools to send data about previously marked locations. They also have obstacle avoidance algorithms. We used differently shaped terrains with obstacles to test the marking algorithms. We compared the DFM and BFM to each other. Additionally, we used a combination of these algorithms. All the marking algorithms perform almost the same. However, the larger the swarm becomes the more efficient the terrain's marking gets.

The last part of the nature-inspired swarm algorithm we dedicate to movement (Chapter 6). The robots within the swarm are decentralized, which means that no central unit is controlling them. However, the robots are within the swarm. To keep them in, we have to define rules, so chaos is not established. For the movement, this is important because the swarm must move in unity, and the robots cannot be separated. Therefore, we defined a three-zone model that takes care of the robots within the swarm. The distance between neighbors is defined for each robot and three zones are defined based on these distances. The first zone is when robots are too close. Therefore, the robot needs to reduce its speed to increase its distance from the others. The second zone is the optimal zone. Having neighborhood robots in this zone means keeping velocity. However, if all the neighborhood is in the third zone, it is critical. In this zone, the robot needs to act fast and gain velocity. Otherwise, the robot will leave the swarm and get lost. We experimented with different widths in the three-zone and their effects on the flock and concluded that changes in the width would affect the flocking behavior of a robot swarm. The last algorithm for the movement module is about phototaxis. Phototaxis is when entities start to move whenever they sense light. We used it on a battery-powered and self-charging robot swarm to see if they would go against the light direction without feeling the emission direction. The experiment aims to stay connected with each other like a swarm and that each robot that senses light moves. The battery-powered ones will move immediately, and the self-charging ones– after they got charged, i.e., after a few seconds. We compare them to each other to find out which mimics the phototactic movement most. The aim was that the robot swarm must go over the finish line within a given time, and the finish line is positioned in the opposite direction of the light emission. We discovered that the self-charging robot swarm mimics the phototactic behavior much better because the robot's movement was not chaotic, which is the success reason.

With the help of the simulator Swarm-Sim, we found that our algorithms for robot swarms could solve three application fields entirely and efficiently. Nevertheless, the algorithms were tested and evaluated in the simulator Swarm-Sim. Therefore, they were not tested on actual robots in real-life scenarios, which is one of the topics of the next chapter, the Future Work.

7.2 Future Work

The simulator scenario had optimal conditions, which means no friction, gravity, weather changes, etc., were used. Next, the robot swarm was tested independently and without any swarm competitors. Therefore, the limitation of the nature-inspired algorithms is that they have not been tested with predictable and well-modeled influencing factors. As a result, one point for future work for Swarm-Sim is to create an environment with additional circumstances such as physical forces and rules.

For the leader coating algorithm, the coordinates of the accessible location were given from the simulator. Thus, the position had no faults. However, in real-life scenarios, sometimes coordinates are faulty, or positions are wrong, e.g., false GPS signals or the coated robots might slip or move from their place, thus, causing the leader to not position all the others correctly. Therefore, it is necessary to add circumstances for such scenarios in future work. Further, the leader-based coating algorithm is limited to simple and cave-shaped objects. It should be updated to coat multiple caves, cave in caves, or caves with bigger diameters for future work. Last, it can be improved by using more than one leader. Another future work point considers the testing of the general coating algorithm. For testing its robustness, we deleted robots from the swarm. However, a defected robot cannot disappear in a real-life environment, and therefore, it is still within the swarm. Thus, an additional algorithm that takes care of the defected robot for future work needs to be developed. To test and evaluate the coating of multiple object islands is interesting. Further, updating the general coating for coating in 3D could be an additional upgrade.

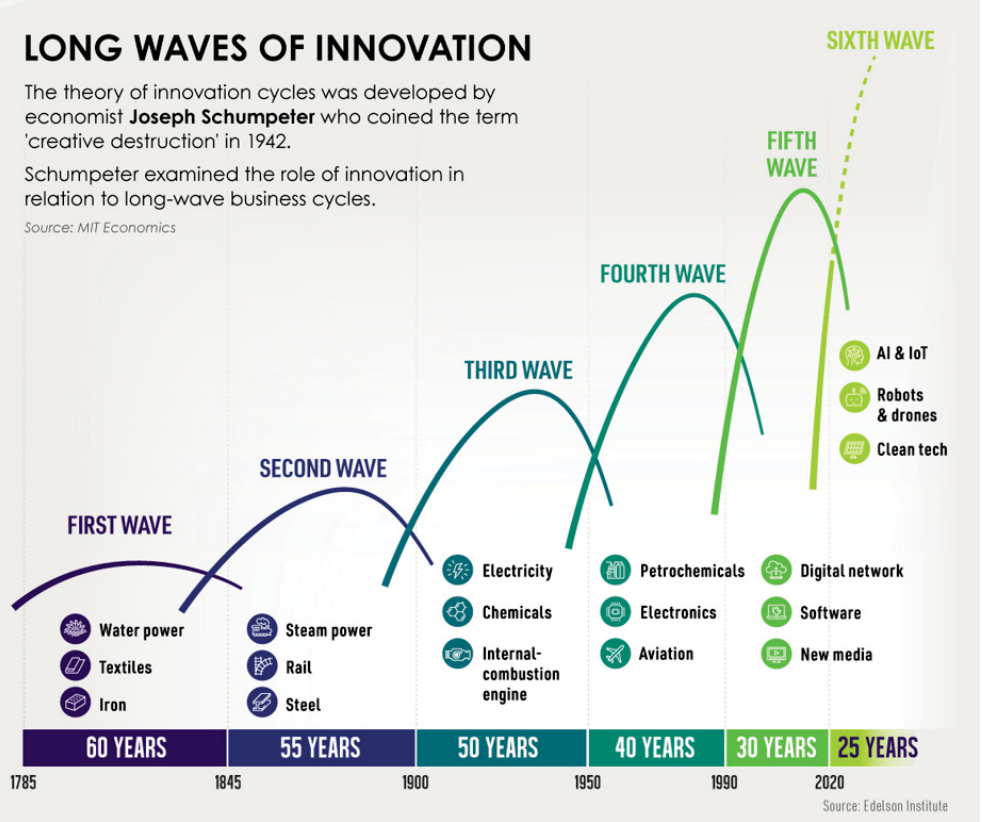
For the OppNet communication within the swarm, only the ProPHET and Epidemic Routing have been tested. There are far more routing protocols for OppNet that can be implemented and simulated. Especially in real-life scenarios, it is vital to see if all those protocols are feasible and can do almost the same. Further, comparing the results with other OppNet simulators, such as ONE simulator, and evaluate memory requirements of delivery probability data structures of PRoPHET. Through the ant communication, we simulated pheromones coming from the ants when they found food, and we implemented protection against ant mills. However, this prevention does not help ants escape from an existing ant mill. Thus, the future work is to implement an ant-mill escaping algorithm, e.g., memorizing visited locations, that allows the ant to escape from the ant mill and not walk till they die. For the OppNet communication within the swarm, only the ProPHET and Epidemic Routing have been tested. There are far more routing protocols for OppNet that can be implemented and simulated. Especially in real-life scenarios, it is vital to see if all these protocols are feasible and can have similar results. Further, comparing the results with other OppNet simulators, such as ONE simulator, and evaluating memory requirements of delivery probability data structures of PRoPHET would be beneficial. Through the ant communication, we simulated ant pheromones when they found food, and we implemented protection against ant-mills. However, this prevention does not help ants escape from an existing ant-mill. Thus, future work can explore implementation of an ant-mill escaping algorithm, e.g., memorizing visited locations, that allows the ant to escape from the ant-mill. Moreover, a malicious ant secure system can be added. It should protect from ants that steal food or spread false pheromones. For the marking algorithm, further research can be done in communication between the robots. The robots can communicate previously explored terrain through marking (indirect communication) or by sharing data (direct communication). With direct communication the robots broadcast the marked

positions. Therefore, other robots other robots do not rediscover terrain because they have received the information that it was already discovered. However, it is would be beneficial to use a filter mechanism or better routing protocols for sharing data to avoid overhead. It could also be helpful to explore new and more efficient marking or obstacles algorithms for future work. For the marking algorithm, the communication of telling that terrain has been observed either through marking (indirect communication) or by sharing data (direct communication). By the second one, the robots broadcast the position that they have been marked. Therefore, other robots prevent discovered terrain by receiving information. However, it is good to use a filter mechanism or better routing protocols for sharing the data to avoid overhead or new and more efficient marking or obstacles algorithms for future work.

In addition, future work could explore fields other than the hexagon field for flocking. The flocking behavior of the robot swarm was only tested in a hexagon field. However, the world does is not hexagonal, it is three-dimensional, and a robot can move in any direction. Therefore, for flocking, it would be of interest in the future to see if this can also work in a three-dimensional field. Another future work for the flocking algorithm is to handle obstacles, so the robots can separate and then form one swarm again. Further, what if the robots are not homogeneous and different in size and characteristics. Can flocking still work? Other than flocking, the phototactic movement of the robot swarm is also explored. For this movement, we only simulate light from one direction. However, how will the swarm behave by having more lights coming from different directions? Further, is it possible to improve the movement using local data, such as the previous location? Moreover, another future work can test the swarm in an environment containing obstacles.

As a result, future work for the algorithm is necessary to adapt to real robots and test within a swarm. However, robots need to be developed that can act as a swarm. Additionally, standards need to be defined that make robot swarm development much more attractive to the science world. The hardware criteria must be standardized, and further, the APIs. Thus, there are plenty of points that need to be solved for natural robot swarms and developing general standardized rules for such circumstances for robot swarms. For the simulator Swarm-Sim, the future work can be to implement add-on scenarios that can affect the simulation, such as disordered coordination systems that cause discrepancy for the robots. Moreover, the rules of physics can be developed. Thus, to simulate forces and seed how those can affect the robot swarm. Further, load balancing can be added to the simulator to make the simulator more scalable. Thus, to increase the size of the robot swarm, and split them into different groups to let them run on other threads.

Consequently, there are many possibilities for improving the algorithm and the simulator, especially for bringing them closer to real-life scenarios. The handling with competitor swarms was not developed and tested in the algorithm. It will be interesting to see, for example how two different swarms try to coat one object, how ants from different nests collect food, or to analyze when more than two swarm flocks face each other. There are still plenty of algorithms that can be developed, such as an algorithm for the swarm to set itself in different forms, such as a cube or hexagon. Techniques of the P2P world can be used within the robot swarm. For example, the robot swarms decentralized count the numbers of robots within the swarm or validate their actions. Therefore, another future work is to use P2P Monitoring [168, 169, 170, 171, 172] algorithms to help robots gain internal information such as the swarm size or broken robots. Also, if the robots could check their current status and validate it, this would be an improvement. Additionally, the storing and sharing of data within robot swarms might



Title: Long Waves: The History of Innovation Cycles
Author: Visual Capitalist
License: Free

Figure 7.1: Innovation Cycle (Source: [181])

be solved with P2P Storage [173, 174, 175, 176, 177]. It helps to reduce overhead and maintain that the data is decentralized, stored, and accessible. Moreover, P2P Applications [178, 179, 180] can be used within the robot swarm as a mobility network farm. To sum up, the future work for robot swarms is to port those algorithms to real robots and test them in a real-life scenario with different conditions. Next, future research can explore new algorithms and find ways to improve the current algorithm to become fault tolerated. So, they can handle flaws and be prepared for obstacles. Thus, artificial intelligence is necessary. The robots should collect data and use those in the future to look out for mistakes or improve their algorithm to be more resistant to faults. The near future will tell us if these developed algorithms will be used for real robots. At least with this dissertation, we have provided a literature survey, a simulator, and seven algorithms for robot swarms for future research.

7.3 Closing Words

The research into nature-inspired robot swarm algorithms is just beginning a new era. Technology in the last decades has been improving continuously, and the sensors have become smaller and cheaper. From the latest innovation cycle shown in Fig. 7.1 we can see that Artificial

Figure 7.2: Altstadtfest (Old Town Festival) celebrating the opening of the rebuilt old town with a drone show (Source: [182])

Intelligence (AI), Internet of things (IoT), robots/drones, and cleantech are innovations that will lead to the next big technology wave. What is missing are robot swarms. However, getting the robots together as a swarm to become more efficient in solving problems might be a part of the upcoming innovation cycle wave. After almost 32 years since the first research into using nature swarm ability in robots, robot swarms are still not mainstream, and we do not see them in our daily life. Today we have LTE, 5G internet, which allows for the monitoring of gadgets. 5G technology allows more bandwidth with low latency, which will allow for central communication and, therefore, make the monitoring of the robots, drones, or even robot swarms much faster and better. AI is now standard, and it will help the robot swarm to become more innovative. Producing hardware has become much cheaper, sensors are getting smaller and more accurate; more possibilities are given with the IoT. Thus, it is time to develop a robot that can coordinate with other robots to build a robot swarm. First, it is necessary to define standards for robots' hardware. The next step is to create an open-source operating system (OS) with some common standards and a solid and motivated community. The OS for the robot swarm will be necessary, and it helps hardware developers to focus only on the hardware without also focusing on the software. Therefore, standards for developing software and APIs, protocols, etc., are a necessary.

Based on our current knowledge, there is only one innovation for robot swarm, and it is drones shows (Fig. 7.2). The drones have a collaboration behavior to form different figures and change their lighting in real-time. However, robot swarms can be used in other future fields such as health care or space science. Nevertheless, all those innovations are still dependent on standards and software. A task force of hardware and software developers needs to be

developed. Additionally, an innovation ground needs to be settled for the robot swarms. Application fields for robot swarms have been presented in chapter 2. However, innovation must define its benefits and how it can generate revenue. The benefits can include cost- or work-efficiency, which makes it attractive for the customer. Therefore, much pre-work and research still must be done to make robot swarms innovative and pleasing to the market.

We provide the simulator Swarm-Sim and all eight algorithms making a tiny drop in the enormous ocean of the robot swarm science and research field. When we see soon that most of the algorithms suggested in this thesis have been implemented for natural robot swarms, we might reflect that this dissertation provided some of the fundamentals bases of the future robot swarm technology that will become normal, like smartphones or digital networks, in the future.

We have the year 2022, and the world is facing a massive increase of startups focusing on building artificial intelligence systems and robot technologies. Tesla presented its robot Optimus [183], which helps us do work that we do not like or assist us in being efficient. Facebook changes its name to Meta to build up the Metaverse so we can do our work and be social in a virtual world. The world is changing, and many science fiction predictions are coming true. So hopefully, soon, our provided algorithms will be used to build robot swarms that help us become more productive in our daily life, heal us from diseases, or protect us against enemies.

Bibliography

[1] Christoph Burgstedt. *3D imaging of antibodies attacking viral cells in the bloodstream*. https://www.shutterstock.com/image-illustration/3d-illustration-antibodies-attacking-virus-cell-1149100028 (Page: 2).

[2] dotun55 via Flickr. *ant trail*. https://www.flickr.com/photos/dotun55/11399922724 (Page: 3).

[3] aquapix. *Shoal of colorful fish in the tropical coral reef*. URL: \url{https://www.shutterstock.com/image-photo/shoal-colorful-fish-tropical-coral-reef-374328919} (Page: 4).

[4] Erol Şahin, Sertan Girgin, Levent Bayindir, and Ali Emre Turgut. "Swarm robotics". In: *Swarm intelligence*. Springer, 2008, pp. 87–100 (Pages: 3, 23).

[5] Erol Sahin, Thomas H Labella, Vito Trianni, J-L Deneubourg, Philip Rasse, Dario Floreano, Luca Gambardella, Francesco Mondada, Stefano Nolfi, and Marco Dorigo. "SWARM-BOT: Pattern formation in a swarm of self-assembling mobile robots". In: *IEEE International Conference on Systems, Man and Cybernetics*. Vol. 4. IEEE. 2002, 6–pp (Pages: 3, 25).

[6] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. "Kilobot: A low cost scalable robot system for collective behaviors". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 3293–3298. DOI: 10.1109/ICRA.2012.6224638 (Page: 6).

[7] Ahmad Reza Cheraghi, Sahdia Shahzad, and Kalman Graffi. *Past, Present, and Future of Swarm Robotics*. 2021, pp. 190–233 (Pages: 8, 9, 17, 21, 23, 25, 26, 97).

[8] Ahmad Cheraghi, Tobias Amft, Salem Sati, Philipp Hagemeister, and Kalman Graffi. "The state of simulation tools for p2p networks on mobile ad-hoc and opportunistic networks". In: *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2016, pp. 1–7 (Pages: 8, 10, 32, 97).

[9] Ahmad Reza Cheraghi, Karol Actun, Sahdia Shahzad, and Kalman Graffi. "Swarm-Sim: A 2D & 3D Simulation Core for Swarm Agents". In: *3rd Int. Conf. of Intelligent Robotic and Control Engineering (IRCE 2020)*. 2020 (Pages: 8, 10, 13, 26, 35–41, 97).

[10] Ahmad Reza Cheraghi and Kalman Graffi. "A Leader Based Coating Algorithm for Simple and Cave Shaped Objects with Robot Swarms". In: *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE. 2020, pp. 43–51 (Pages: 8, 26, 43–49, 97).

[11] Ahmad Reza Cheraghi, Gorden Wunderlich, and Kalman Graffi. "General Coating of Arbitrary Objects Using Robot Swarms". In: *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE. 2020, pp. 59–67 (Pages: 8, 26, 52–58, 97).

[12] Ahmad Reza Cheraghi, Julian Zenz, and Kalman Graffi. "Opportunistic Network Behavior in a Swarm: Passing Messages to Destination". In: *Proceedings of the 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE. 2020, pp. 138–144 (Pages: 8, 26, 61, 63–65, 97).

[13] Ahmad Reza Cheraghi, Jochen Peters, and Kalman Graffi. "Prevention of Ant Mills in Pheromone-Based Search Algorithm for Robot Swarms". In: *Submitted to 3rd Int. Conf. of Intelligent Robotic and Control Engineering (IRCE'20)*. 2020 (Pages: 8, 26, 68, 69, 71–73, 97).

[14] Ahmad Reza Cheraghi, Abdelrahman Abdelgalil, and Kalman Graffi. "Universal 2-Dimensional Terrain Marking for Autonomous Robot Swarms". In: *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE. 2020, pp. 24–32 (Pages: 8, 26, 75–78, 97).

[15] Ahmad Reza Cheraghi, Asma Ben Janete, and Kalman Graffi. "Robot Swarm Flocking on a 2D Triangular Graph". In: *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE. 2020, pp. 154–162 (Pages: 8, 26, 82, 83, 85, 86, 97).

[16] Ahmad Reza Cheraghi, Fabio Schloesser Vila, and Kalman Graffi. "Phototactic Movement of Battery-Powered and Self-Charging Robot Swarms". In: *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE. 2020, pp. 73–79 (Pages: 8, 26, 88, 89, 91, 92, 97).

[17] Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W Richa, Christian Scheideler, and Thim Strothmann. "Amoebot-a new model for programmable matter". In: *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*. 2014, pp. 220–222 (Pages: 10, 31, 35, 42).

[18] Zahra Derakhshandeh, Robert Gmyr, Andréa W Richa, Christian Scheideler, and Thim Strothmann. "Universal shape formation for programmable matter". In: *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*. 2016, pp. 289–299 (Pages: 10, 31, 35, 42).

[19] G. Beni. "The concept of cellular robotic system". In: *Proceedings IEEE International Symposium on Intelligent Control 1988*. 1988, pp. 57–62. DOI: 10.1109/ISIC.1988.65405 (Page: 18).

[20] Toshio Fukuda and Seiya Nakagawa. "Approach to the dynamically reconfigurable robotic system". In: *Journal of Intelligent and Robotic Systems* 1.1 (1988), pp. 55–72 (Page: 18).

[21] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. "A taxonomy for swarm robots". In: *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*. Vol. 1. 1993, 441–447 vol.1. DOI: 10.1109/IROS.1993.583135 (Page: 18).

[22] Maja J Mataric. "Designing emergent behaviors: From local interactions to collective intelligence". In: *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. 1993, pp. 432–441 (Page: 18).

[23] Maja J Matarić. "Issues and approaches in the design of collective autonomous agents". In: *Robotics and autonomous systems* 16.2-4 (1995), pp. 321–331 (Pages: 18, 20).

[24] Jean-Louis Deneubourg, Simon Goss, Nigel Franks, Ana Sendova-Franks, Claire Detrain, and Laeticia Chrétien. "The dynamics of collective sorting robot-like ants and ant-like robots". In: *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*. 1991, pp. 356–363 (Page: 18).

[25] Douglas W Gage. *Command control for many-robot systems*. Technical Report: Naval Command Control, Ocean Surveillance Center Rdt And E Div San Diego CA, 1992 (Page: 18).

[26] C Ronald Kube and Eric Bonabeau. "Cooperative transport by ants and robots". In: *Robotics and autonomous systems* 30.1-2 (2000), pp. 85–101 (Page: 18).

[27] Gerardo Beni. "From swarm intelligence to swarm robotics". In: *International Workshop on Swarm Robotics*. Springer. 2004, pp. 1–9 (Pages: 18, 19, 57).

[28] Daniel B Kearns. "A field guide to bacterial swarming motility". In: *Nature Reviews Microbiology* 8.9 (2010), p. 634 (Page: 18).

[29] Cambridge University Press 2021. *intelligence*. 2021. URL: https://dictionary.cambridge.org/dictionary/english/intelligence (visited on 11/2021) (Page: 18).

[30] G Beni and J Wang. *Swarm Intelligence (Proceedgins Seventh Annual Meeting of the Robotics Society of Japan)*. 1989 (Page: 19).

[31] Sanza T Kazadi. "Swarm engineering". PhD thesis. California Institute of Technology, 2000 (Page: 19).

[32] Jan Carlo Barca and Y Ahmet Sekercioglu. "Swarm robotics reviewed". In: *Robotica* 31.3 (2013), pp. 345–359 (Page: 20).

[33] Belkacem Khaldi and Foudil Cherif. "An overview of swarm robotics: Swarm intelligence applied to multi-robotics". In: *International Journal of Computer Applications* 126.2 (2015) (Page: 20).

[34] Y Tan. "Swarm robotics: collective behavior inspired by nature". In: *J Comput Sci Syst Biol* 6 (2013), e106 (Page: 20).

[35] Ying Tan and Zhong-yang Zheng. "Research advance in swarm robotics". In: *Defence Technology* 9.1 (2013), pp. 18–39 (Pages: 20, 23).

[36] Erol Şahin. "Swarm robotics: From sources of inspiration to domains of application". In: *International workshop on swarm robotics*. Springer. 2004, pp. 10–20 (Page: 20).

[37] Vito Trianni, Elio Tuci, Christos Ampatzis, and Marco Dorigo. "Evolutionary swarm robotics: A theoretical and methodological itinerary from individual neuro-controllers to collective behaviours". In: *The horizons of evolutionary robotics* 153 (2014) (Page: 20).

[38] Paul M Maxim, William M Spears, and Diana F Spears. "Robotic chain formations". In: *IFAC Proceedings Volumes* 42.22 (2009), pp. 19–24 (Page: 20).

[39] Marco Dorigo. "SWARM-BOT: An experiment in swarm robotics". In: *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*. IEEE. 2005, pp. 192–200 (Page: 20).

[40] Marco Dorigo, Elio Tuci, Vito Trianni, Roderich Gro, Shervin Nouyan, Christos Ampatzis, Thomas Labella, Rehan O'Grady, Michael Bonani, and Francesco Mondada. "SWARM-BOT: Design and implementation of colonies of self-assembling robots". In: Jan. 2006, pp. 103–135 (Page: 20).

[41] Marco Dorigo, Vito Trianni, Erol Şahin, Roderich Groß, Thomas H Labella, Gianluca Baldassarre, Stefano Nolfi, Jean-Louis Deneubourg, Francesco Mondada, Dario Floreano, et al. "Evolving self-organizing behaviors for a swarm-bot". In: *Autonomous Robots* 17.2-3 (2004), pp. 223–245 (Pages: 20, 25).

[42] Onur Soysal, Erkin Bahçeci, and Erol Şahin. "Aggregation in swarm robotic systems: Evolution and probabilistic control". In: *Turkish Journal of Electrical Engineering & Computer Sciences* 15.2 (2007), pp. 199–225 (Page: 20).

[43] Onur Soysal and Erol Sahin. "Probabilistic aggregation strategies in swarm robotic systems". In: *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*. IEEE. 2005, pp. 325–332 (Page: 20).

[44] Eliseo Ferrante, Ali Emre Turgut, Cristián Huepe, Alessandro Stranieri, Carlo Pinciroli, and Marco Dorigo. "Self-organized flocking with a mobile robot swarm: a novel motion control method". In: *Adaptive Behavior* 20.6 (2012), pp. 460–477 (Page: 20).

[45] Adam T Hayes and Parsa Dormiani-Tabatabaei. "Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots". In: *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. Vol. 4. IEEE. 2002, pp. 3900–3905 (Page: 20).

[46] M. Masár. "A biologically inspired swarm robot coordination algorithm for exploration and surveillance". In: *2013 IEEE 17th International Conference on Intelligent Engineering Systems (INES)*. 2013, pp. 271–275. DOI: 10.1109/INES.2013.6632825 (Page: 20).

[47] Wenguo Liu and Alan Winfield. "Modeling and Optimization of Adaptive Foraging in Swarm Robotic Systems". In: *I. J. Robotic Res.* 29 (Dec. 2010), pp. 1743–1760. DOI: 10.1177/0278364910375139 (Page: 20).

[48] Shervin Nouyan, Alexandre Campo, and Marco Dorigo. "Path formation in a robot swarm". In: *Swarm Intelligence* 2.1 (2008), pp. 1–23 (Page: 21).

[49] Frederick Ducatelle, Gianni A Di Caro, Carlo Pinciroli, Francesco Mondada, and Luca Gambardella. "Communication assisted navigation in robotic swarms: self-organization and cooperation". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE. 2011, pp. 4981–4988 (Page: 21).

[50] Roderich Groβ and Marco Dorigo. "Evolution of solitary and group transport behaviors for autonomous robots capable of self-assembling". In: *Adaptive Behavior* 16.5 (2008), pp. 285–305 (Page: 21).

[51] Mark Trueblood and Russell Genet. "Microcomputer control of telescopes". In: *Richmond: Willmann-Bell, 1985* (1985) (Page: 21).

[52] Alasdair Allan, Tim Naylor, Iain Steele, Dave Carter, Tim Jenness, Frossie Economou, and Andy Adamson. "eSTAR: Astronomers, Agents and when Robotic Telescopes aren't..." In: *Astronomical Data Analysis Software and Systems (ADASS) XIII*. Vol. 314. 2004, p. 597 (Pages: 21, 27).

[53] Cindy Mason. "Collaborative Networks of Independent Automatic Telescopes". In: *Optical Astronomy from the Earth and Moon*. Vol. 55. 1994, p. 234 (Page: 21).

[54] John EF Baruch. "Robots in astronomy". In: *Vistas in Astronomy* 35 (1992), pp. 399–438 (Pages: 21, 27).

[55] Vito Trianni, Joris IJsselmuiden, and Ramon Haken. *The Saga Concept: Swarm Robotics for Agricultural Applications*. Technical Report: Technical Report. 2016. Available online: http://laral. istc. cnr. it/saga ..., 2016 (Pages: 21, 24).

[56] Dario Albani, Joris IJsselmuiden, Ramon Haken, and Vito Trianni. "Monitoring and mapping with robot swarms for agricultural applications". In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2017, pp. 1–6 (Pages: 21, 24).

[57] *SAGA Swarm Robotics for Aggriculture Applications*. http://laral.istc.cnr.it/saga/ (Pages: 21, 24).

[58] Simon Blackmore. "Precision farming: an introduction". In: *Outlook on agriculture* 23.4 (1994), pp. 275–280 (Page: 21).

[59] *Avular howpublished = https://www.avular.com* (Page: 21).

[60] *Drones.nl*. https://www.drones.nl/bedrijven/avular (Page: 21).

[61] *Swarm Farm Robotic Agriculture* (Page: 21).

[62] Markus Kayser, Levi Cai, Christoph Bader, Sara Falcone, Nassia Inglessis, Barrak Darweesh, João Costa, and Neri Oxman. "FIBERBOTS: Design and Digital Fabrication of Tubular Structures Using Robot Swarms". In: *Robotic Fabrication in Architecture, Art and Design*. Springer. 2018, pp. 285–296 (Pages: 21, 25).

[63] *Alibab's Flyzoo Future Hotel*. https://www.alizila.com/introducing-alibabas-flyzoo-future-hotel/ (Page: 21).

[64] Víctor García-López, Fang Chen, Lizanne G Nilewski, Guillaume Duret, Amir Aliyan, Anatoly B Kolomeisky, Jacob T Robinson, Gufeng Wang, Robert Pal, and James M Tour. "Molecular machines open cell membranes". In: *Nature* 548.7669 (2017), p. 567 (Page: 21).

[65] *Millirobot with a talent for versatility of movement*. https://www.mpg.de/11895964/millirobot- (Pages: 21, 27).

[66] *Nature-inspired soft millirobot makes its way through enclosed spaces* (Pages: 21, 27).

[67] *Biorobotics Laboratory*. http://biorobotics.ri.cmu.edu/robots/index.php (Pages: 21, 27).

[68] *Trunk Snake Robot*. http://biorobotics.ri.cmu.edu/robots/trunkSnake.php (Page: 21).

[69] *Medical Snake Robot*. http://biorobotics.ri.cmu.edu/robots/medSnake.php (Page: 21).

[70] *Fullabot*. http://biorobotics.ri.cmu.edu/robots/fullabot.php (Page: 21).

[71] *Endeavor Robotics*. http://endeavorrobotics.com/products (Pages: 21, 27).

[72] *PackBot*. https://robots.ieee.org/robots/packbot/ (Pages: 21, 27).

[73] *LS3 Legged Squad Support Systems*. https://www.bostondynamics.com/ls3 (Pages: 21, 27).

[74] Armin Krishnan. *Killer robots: legality and ethicality of autonomous weapons*. Routledge, 2016 (Page: 21).

[75] Stuart Young and Alexander Kott. "A survey of research on control of teams of small robots in military operations". In: *arXiv preprint arXiv:1606.01288* (2016) (Page: 21).

[76] *Flying mini-robot cleaners win Electrolux Design Lab 2013 Contest* (Pages: 21, 27).

[77] Abhishek Gupta, Akash Saxena, Prasun Anand, Pooja Sharma, Prince Raj Goyal, and Ranjeet Singh. "Robo-Cleaner". In: *Imperial Journal of Interdisciplinary Research* 2.5 (2016). ISSN: 2454-1362. URL: http://www.imperialjournals.com/index.php/IJIR/article/view/457 (Page: 21).

[78] Veerajagadheswar Prabakaran, Mohan Rajesh Elara, Thejus Pathmakumar, and Shunsuke Nansai. "Floor cleaning robot with reconfigurable mechanism". In: *Automation in Construction* 91 (2018), pp. 155–165 (Page: 21).

[79] Abhishek Pandey, Anirudh Kaushik, Amit Kumar Jha, and Girish Kapse. "A Technological Survey on Autonomous Home Cleaning Robots". In: *International Journal of Scientific and Research Publications* 4.4 (2014), pp. 1–7 (Page: 21).

[80] Paolo Fiorini and Erwin Prassler. "Cleaning and household robots: A technology survey". In: *Autonomous robots* 9.3 (2000), pp. 227–235 (Page: 21).

[81] Andrey Ronzhin, Gerhard Rigoll, and Roman Meshcheryakov. *Interactive Collaborative Robotics: Third International Conference, ICR 2018, Leipzig, Germany, September 18–22, 2018, Proceedings*. Vol. 11097. Springer, 2018 (Page: 22).

[82] Pattie Maes. "Artificial life meets entertainment: lifelike autonomous agents". In: *Communications of the ACM* 38.11 (1995), pp. 108–114 (Page: 22).

[83] Levent Bayindir and Erol Şahin. "A review of studies in swarm robotics". In: *Turkish Journal of Electrical Engineering & Computer Sciences* 15.2 (2007), pp. 115–147 (Page: 23).

[84] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. "Swarm robotics: a review from the swarm engineering perspective". In: *Swarm Intelligence* 7.1 (2013), pp. 1–41 (Page: 23).

[85] Francesco Mondada, André Guignard, Michael Bonani, Daniel Bar, Michel Lauria, and Dario Floreano. "Swarm-bot: From concept to implementation". In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*. Vol. 2. IEEE. 2003, pp. 1626–1631 (Page: 25).

[86] Francesco Mondada, Giovanni C Pettinaro, Andre Guignard, Ivo W Kwee, Dario Floreano, Jean-Louis Deneubourg, Stefano Nolfi, Luca Maria Gambardella, and Marco Dorigo. "SWARM-BOT: A new distributed robotic concept". In: *Autonomous robots* 17.2-3 (2004), pp. 193–221 (Page: 25).

[87] Marco Dorigo, Elio Tuci, Roderich Groß, Vito Trianni, Thomas Halva Labella, Shervin Nouyan, Christos Ampatzis, Jean-Louis Deneubourg, Gianluca Baldassarre, Stefano Nolfi, et al. "The swarm-bots project". In: *International Workshop on Swarm Robotics*. Springer. 2004, pp. 31–44 (Page: 25).

[88] Marco Dorigo. "Swarm-Bots and Swarmanoid: Two Experiments in Embodied Swarm Intelligence." In: *Web intelligence*. 2009, pp. 2–3 (Page: 25).

[89] Carlo Pinciroli. "The swarmanoid simulator". In: *Bruxelles: UniversitéLibre de Bruxelles* (2007) (Pages: 25, 26).

[90] Frederick Ducatelle, Gianni A Di Caro, Carlo Pinciroli, and Luca M Gambardella. "Self-organized cooperation between robotic swarms". In: *Swarm Intelligence* 5.2 (2011), p. 73 (Page: 25).

[91] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, et al. "Swarmanoid: a novel concept for the study of heterogeneous robotic swarms". In: *IEEE Robotics & Automation Magazine* 20.4 (2013), pp. 60–71 (Page: 25).

[92] Antoine Decugniere, Benjamin Poulain, Alexandre Campo, Carlo Pinciroli, Bruno Tartini, Michel Osee, Marco Dorigo, and Mauro Birattari. "The cart-bot and the cooperative transport of multiple objects in the swarmanoid project". In: *Technical Report TR/IRIDIA/2008–014 IRIDIA, Universite Libre de Bruxelles* (2008) (Page: 25).

[93] Heinz Woern, Marc Szymanski, and Joerg Seyfried. "The i-swarm project". In: *RO-MAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE. 2006, pp. 492–496 (Page: 25).

[94] Jörg Seyfried, Marc Szymanski, Natalie Bender, Ramon Estaña, Michael Thiel, and Heinz Wörn. "The I-SWARM project: Intelligent small world autonomous robots for micro-manipulation". In: *International Workshop on Swarm Robotics*. Springer. 2004, pp. 70–83 (Page: 25).

[95] Aveek Purohit, Frank Mokaya, and Pei Zhang. "Demo abstract: Collaborative indoor sensing with the SensorFly aerial sensor network". In: *2012 ACM/IEEE 11th International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE. 2012, pp. 145–146 (Page: 25).

[96] Aveek Purohit, Zheng Sun, Frank Mokaya, and Pei Zhang. "SensorFly: Controlled-mobile sensing platform for indoor emergency response applications". In: *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE. 2011, pp. 223–234 (Page: 25).

[97] James E Bluman, Chang-Kwon Kang, D Brian Landrum, Farbod Fahimi, and Bryan Mesmer. "Marsbee-Can a Bee Fly on Mars?" In: *55th AIAA Aerospace Sciences Meeting*. 2017, p. 0328 (Page: 25).

[98] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. "Kilobot: A low cost scalable robot system for collective behaviors". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 3293–3298 (Page: 25).

[99] Gabriele Valentini, Anthony Antoun, Marco Trabattoni, Bernát Wiandt, Yasumasa Tamura, Etienne Hocquard, Vito Trianni, and Marco Dorigo. "Kilogrid: a novel experimental environment for the kilobot robot". In: *Swarm Intelligence* 12.3 (2018), pp. 245–266 (Page: 25).

[100] Fredrik Jansson, Matthew Hartley, Martin Hinsch, Ivica Slavkov, Noemí Carranza, Tjelvar SG Olsson, Roland M Dries, Johanna H Grönqvist, Athanasius FM Marée, James Sharpe, et al. "Kilombo: a Kilobot simulator to enable effective research in swarm robotics". In: *arXiv preprint arXiv:1511.04285* (2015) (Page: 25).

[101] Yuri K Lopes, André B Leal, Tony J Dodd, and Roderich Groß. "Application of supervisory control theory to swarms of e-puck and kilobot robots". In: *International Conference on Swarm Intelligence*. Springer. 2014, pp. 62–73 (Page: 25).

[102] Michael Rubenstein and Radhika Nagpal. "Kilobot: A Robotic Module for Demonstrating Behaviors in a Large Scale (\(2^{10}\) Units) Collective". In: Institute of Electrical and Electronics Engineers. 2010 (Page: 25).

[103] Ali E Turgut, F Gokce, Hande Celikkanat, L Bayindir, and Erol Sahin. "Kobot: A mobile robot designed specifically for swarm robotics research". In: *Middle East Technical University, Ankara, Turkey, METU-CENG-TR Tech. Rep* 5.2007 (2007) (Page: 25).

[104] *Brian Gerkey's website*. https://brian.gerkey.org (Page: 26).

[105] Richard T Vaughan and Brian P Gerkey. "Reusable robot software and the player/stage project". In: *Software Engineering for Experimental Robotics*. Springer, 2007, pp. 267–289 (Page: 26).

[106] Brian P Gerkey, Richard T Vaughan, Kasper Stoy, Andrew Howard, Gaurav S Sukhatme, and Maja J Mataric. "Most valuable player: A robot device server for distributed control". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the Next Millennium (Cat. No. 01CH37180)*. Vol. 3. IEEE. 2001, pp. 1226–1231 (Page: 26).

[107] Brian Gerkey, Richard T Vaughan, and Andrew Howard. "The player/stage project: Tools for multi-robot and distributed sensor systems". In: *Proceedings of the 11th international conference on advanced robotics*. Vol. 1. 2003, pp. 317–323 (Page: 26).

[108] Richard Vaughan. "Massively multi-robot simulation in stage". In: *Swarm intelligence* 2.2-4 (2008), pp. 189–208 (Page: 26).

[109] *Open Robotics*. https://www.openrobotics.org (Page: 26).

[110] *Robot Operating System*. http://www.ros.org (Page: 26).

[111] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar Von Stryk. "Comprehensive simulation of quadrotor uavs using ros and gazebo". In: *International conference on simulation, modeling, and programming for autonomous robots*. Springer. 2012, pp. 400–411 (Page: 26).

[112] Nathan Koenig and Andrew Howard. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE. 2004, pp. 2149–2154 (Page: 26).

[113] *Robomatter Incorporated*. http://www.robomatter.com (Page: 26).

[114] *Robotc*. http://www.robotc.net (Page: 26).

[115] *Robot Virtual Worlds*. http://www.robotvirtualworlds.com (Page: 26).

[116] *Georgia Tech's Mobile Robot Laboratory*. https://www.cc.gatech.edu/ai/robot-lab/ (Page: 26).

[117] T Balch. "TeamBots software and documentation". In: *Available through the World-Wide Web at http://www. teambots. org* (2001) (Page: 26).

[118] Tucker Balch. "The TeamBots Environment for Multi-Robot Systems Development". In: *Working notes of Tutorial on Mobile Robot Programming Paradigms, ICRA* (2002) (Page: 26).

[119] Tucker Balch. *Behavioral diversity in learning robot teams*. Technical Report: Georgia Institute of Technology, 1998 (Page: 26).

[120] *V-REP*. http://www.coppeliarobotics.com (Page: 26).

[121] Marc Freese, Surya Singh, Fumio Ozaki, and Nobuto Matsuhira. "Virtual robot experimentation platform v-rep: A versatile 3d robot simulator". In: *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer. 2010, pp. 51–62 (Page: 26).

[122] Miguel A Olivares-Mendez, Somasundar Kannan, and Holger Voos. "Vision based fuzzy control autonomous landing with UAVs: From V-REP to real experiments". In: *2015 23rd Mediterranean Conference on Control and Automation (MED)*. IEEE. 2015, pp. 14–21 (Page: 26).

[123] E Peralta, E Fabregas, G Farias, H Vargas, and S Dormido. "Development of a Khepera IV Library for the V-REP Simulator". In: *IFAC-PapersOnLine* 49.6 (2016), pp. 81–86 (Page: 26).

[124] Eric Rohmer, Surya PN Singh, and Marc Freese. "V-REP: A versatile and scalable robot simulation framework". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 1321–1326 (Page: 26).

[125] Stéphane Magnenat, Philippe Rétornaz, Michael Bonani, Valentin Longchamp, and Francesco Mondada. "ASEBA: A modular architecture for event-based control of complex robots". In: *IEEE/ASME transactions on mechatronics* 16.2 (2011), pp. 321–329 (Page: 26).

[126] M Allwright, N Bhalla, C Pinciroli, and M Dorigo. *ARGoS plug-ins for experiments in autonomous construction*. Technical Report: Technical report TR/IRIDIA/2018-007, IRIDIA, Université Libre de Bruxelles ..., 2018 (Page: 26).

[127] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. "ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 5027–5034 (Page: 26).

[128] *Webots Open Source Robot Simulator*. https://cyberbotics.com (Page: 26).

[129] Olivier Michel. "Webots: Symbiosis between virtual and real mobile robots". In: *International Conference on Virtual Worlds*. Springer. 1998, pp. 254–263 (Page: 26).

[130] Michel Olivier. "Cyberbotics LTD-webotstm: Professional mobile robot simulation". In: *International Journal of Advanced Robotic Systems* 1.1 (2004), pp. 40–43 (Page: 26).

[131] LF Wang, KC Tan, and V Prahlad. "Developing Khepera robot applications in a Webots environment". In: *MHS2000. Proceedings of 2000 International Symposium on Micromechatronics and Human Science (Cat. No. 00TH8530)*. IEEE. 2000, pp. 71–76 (Page: 26).

[132] *Watsolutions*. http://www.watsolutions.com (Page: 26).

[133] *Work space ROBOT SIMULATION*. http://www.workspace5.com (Page: 26).

[134] *AIST*. https://www.aist.go.jp/index_en.html (Page: 26).

[135] Fumio Kanehiro, Hirohisa Hirukawa, and Shuuji Kajita. "OpenHRP: Open architecture humanoid robotics platform". In: *The International Journal of Robotics Research* 23.2 (2004), pp. 155–165 (Page: 26).

[136] Rajat Mittal, Atsushi Konno, and Shunsuke Komizunai. "Implementation of HOAP-2 humanoid walking motion in openHRP simulation". In: *2015 International Conference on Computing Communication Control and Automation*. IEEE. 2015, pp. 29–34 (Page: 26).

[137] Hirohisa Hirukawa, Fumio Kanehiro, Kenji Kaneko, Shuuji Kajita, Kiyoshi Fujiwara, Yoshihiro Kawai, Fumiaki Tomita, Shigeoki Hirai, Kazuo Tanie, Takakatsu Isozumi, et al. "Humanoid robotics platforms developed in HRP". In: *Robotics and Autonomous Systems* 48.4 (2004), pp. 165–175 (Page: 26).

[138] Kevin DeMarco, Eric Squires, Michael Day, and Charles Pippin. "Simulating collaborative robots in a massive multi-agent game environment (SCRIMMAGE)". In: *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 283–297 (Page: 26).

[139] Craig W Reynolds. "Flocks, herds and schools: A distributed behavioral model". In: *ACM SIGGRAPH computer graphics*. Vol. 21. 4. ACM. 1987, pp. 25–34 (Page: 25).

[140] Darren Quick. *Honda sets its sights on an eVTOL, telepresence robot and space tech.* 2021. URL: https://HondasetsitssightsonaneVTOL,telepresencerobotandspacetech.com/technology/honda-future-focus-evtol-telepresence-robot-space/ (Page: 28).

[141] Matthias Feldotto and Kalman Graffi. "Systematic evaluation of peer-to-peer systems using PeerfactSim. KOM". In: *Concurrency and Computation: Practice and Experience* 28.5 (2016), pp. 1655–1677 (Page: 31).

[142] Matthias Feldotto and Kalman Graffi. "Comparative evaluation of peer-to-peer systems using PeerfactSim. KOM". In: *2013 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE. 2013, pp. 99–106 (Page: 31).

[143] Aleksandra Kovacevic, Sebastian Kaune, Hans Heckel, André Mink, Kalman Graffi, Oliver Heckmann, and Ralf Steinmetz. "PeerfactSim. KOM-A Simulator for Large-Scale Peer-to-Peer Networks". In: *Technische Universität Darmstadt, Germany, Tech. Rep. Tr-2006-06* (2006) (Page: 31).

[144] Kalman Graffi. "PeerfactSim. KOM: A P2P system simulator—Experiences and lessons learned". In: *2011 IEEE International Conference on Peer-to-Peer Computing*. IEEE. 2011, pp. 154–155 (Page: 31).

[145] Markus Benter, Mohammad Divband, Sebastian Kniesburges, Andreas Koutsopoulos, and Kalman Graffi. "Ca-re-chord: A churn resistant self-stabilizing chord overlay network". In: *2013 Conference on Networked Systems*. IEEE. 2013, pp. 27–34 (Page: 32).

[146] Tobias Amft, Barbara Guidi, Kalman Graffi, and Laura Ricci. "FRoDO: Friendly routing over dunbar-based overlays". In: *2015 IEEE 40th conference on local computer networks (LCN)*. IEEE. 2015, pp. 356–364 (Page: 32).

[147] Tobias Amft and Kalman Graffi. *The Benefit of Stacking Multiple Peer-to-Peer Overlays*. Technical Report: Technical Report: TR-2017-002. Technology of Social Networks Group, Heinrich Heine University, 2017 (Page: 32).

[148] Tobias Amft and Kalman Graffi. "A Tale of Many Networks: Splitting and Merging of Chord-like Overlays in Partitioned Networks". In: *Technology of Social Networks Group, Heinrich Heine University, Düsseldorf, Germany, Tech. Rep. TR-2017-001* (2017) (Page: 32).

[149] Tobias Amft and Kalman Graffi. "Moving peers in distributed, location-based peer-to-peer overlays". In: *2017 international conference on computing, networking and communications (ICNC)*. IEEE. 2017, pp. 906–911 (Page: 32).

[150] Parag S Mogre, Kalman Graffi, Matthias Hollick, and Ralf Steinmetz. "AntSec, WatchAnt, and AntRep: Innovative Security Mechanisms for Wireless Mesh Networks". In: *32nd IEEE Conference on Local Computer Networks (LCN 2007)*. IEEE. 2007, pp. 539–547 (Page: 32).

[151] Parag S Mogre, Kalman Graffi, Matthias Hollick, and Ralf Steinmetz. "A security framework for wireless mesh networks". In: *Wireless Communications and Mobile Computing* 11.3 (2011), pp. 371–391 (Page: 32).

[152] Kalman Graffi, Parag S Mogre, Matthias Hollick, and Ralf Steinmetz. "Detection of colluding misbehaving nodes in mobile ad hoc and wireless mesh networks". In: *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*. IEEE. 2007, pp. 5097–5101 (Page: 32).

[153] Tambako The Jaguar. *Lioness carrying her newborn cub*. https://www.flickr.com/photos/tambako/5207804374 (Page: 50).

[154] 12019. *Wolves hunting*. https://pixabay.com/images/id-80497/ (Page: 59).

[155] Salem Sati, Andre Ippisch, and Kalman Graffi. "Dynamic replication control strategy for opportunistic networks". In: *2017 International Conference on Computing, Networking and Communications (ICNC)*. IEEE. 2017, pp. 1017–1023 (Page: 62).

[156] Andre Ippisch and Kalman Graffi. "Infrastructure mode based opportunistic networks on android devices". In: *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*. IEEE. 2017, pp. 454–461 (Page: 62).

[157] Salem Sati and Kalman Graffi. "Adapting the beacon interval for opportunistic network communications". In: *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2015, pp. 6–12 (Page: 62).

[158] Salem Sati, Andre Ippisch, and Kalman Graffi. "Replication probability-based routing scheme for opportunistic networks". In: *2017 International Conference on Networked Systems (NetSys)*. IEEE. 2017, pp. 1–8 (Page: 62).

[159] Andre Ippisch, Salem Sati, and Kalman Graffi. "Device to device communication in mobile delay tolerant networks". In: *2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE. 2017, pp. 1–8 (Page: 62).

[160] Andre Ippisch, Salem Sati, and Kalman Graffi. "Optimal replication based on optimal path hops for opportunistic networks". In: *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*. IEEE. 2018, pp. 251–258 (Page: 62).

[161] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. "The ONE simulator for DTN protocol evaluation". In: *Proceedings of the International Conference on Simulation Tools and Techniques (SIMUtools)*. ICST, 2009 (Page: 65).

[162] Andrey Popov. *Businesspeople Gossiping Behind Stressed Female Colleague In Office*. URL: \url{https://www.shutterstock.com/image-photo/businesspeople-gossiping-behind-stressed-female-colleague-420085351} (Page: 66).

[163] Ching Louis Liu. *dog peeing in the park*. https://www.shutterstock.com/image-photo/dog-peeing-park-544727569 (Page: 79).

[164] Annette Shaff. *border collie barking with a wide open mouth in a studio shot isolated on a blue background*. https://www.shutterstock.com/image-photo/border-collie-barking-wide-open-mouth-1357549616 (Page: 80).

[165] Iain D Couzin, Jens Krause, Richard James, Graeme D Ruxton, and Nigel R Franks. "Collective memory and spatial sorting in animal groups". In: *Journal of theoretical biology* 218.1 (2002), pp. 1–11 (Page: 81).

[166] Pixnio. *Birds flock*. https://pixnio.com/fauna-animals/birds/bird-flock-waterfowl-goose-sea-migration-sky-landscape-water-beach (Page: 87).

[167] *Moths and insects flying around a light globe*. https://www.shutterstock.com/image-photo/moths-insects-flying-around-light-globe-1860081607 (Page: 94).

[168] Vitaliy Rapp and Kalman Graffi. "Continuous gossip-based aggregation through dynamic information aging". In: *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2013, pp. 1–7 (Page: 101).

[169] Andreas Disterhöft and Kalman Graffi. "Convex Hull Watchdog: Mitigation of Malicious Nodes in Tree-Based P2P Monitoring Systems". In: *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE. 2016, pp. 52–60 (Page: 101).

[170] Andreas Disterhöft and Kalman Graffi. "CapSearch: Capacity-Based Search in Highly Dynamic Peer-to-Peer Networks". In: *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*. IEEE. 2017, pp. 621–630 (Page: 101).

[171] Kalman Graffi and Andreas Disterhöft. "SkyEye: A tree-based peer-to-peer monitoring approach". In: *Pervasive and Mobile Computing* 40 (2017), pp. 593–610 (Page: 101).

[172] Andreas Disterhöft, Phillip Sandkühler, Andre Ippisch, and Kalman Graffi. "Mr. Tree: Multiple Realities in Tree-based Monitoring Overlays for Peer-to-Peer Networks". In: *2018 International Conference on Computing, Networking and Communications (ICNC)*. IEEE. 2018, pp. 354–360 (Page: 101).

[173] Philip Wette and Kalman Graffi. "Adding capacity-aware storage indirection to homogeneous distributed hash tables". In: *2013 Conference on Networked Systems*. IEEE. 2013, pp. 35–42 (Page: 102).

[174] Ahmad Rabay'a, Eduard Schleicher, and Kalman Graffi. "Fog computing with p2p: Enhancing fog computing bandwidth for iot scenarios". In: *2019 International Conference on Internet of things (iThings) and IEEE green Computing and communications (GreenCom) and IEEE Cyber, Physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE. 2019, pp. 82–89 (Page: 102).

[175] Jens Janiuk, Alexander Mäcker, and Kalman Graffi. "Secure distributed data structures for peer-to-peer-based social networks". In: *2014 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE. 2014, pp. 396–405 (Page: 102).

[176] Andrea De Salve, Paolo Mori, Laura Ricci, Raed Al-Aaridhi, and Kalman Graffi. "Privacy-preserving data allocation in decentralized online social networks". In: *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, Cham. 2016, pp. 47–60 (Page: 102).

[177] Raed Al-Aaridhi and Kalman Graffi. "Sets, lists and trees: distributed data structures on distributed hash tables". In: *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*. IEEE. 2016, pp. 1–8 (Page: 102).

[178] Newton Masinde and Kalman Graffi. "Peer-to-peer-based social networks: A comprehensive survey". In: *SN Computer Science* 1.5 (2020), pp. 1–51 (Page: 102).

[179] Andreas Disterhoft and Kalman Graffi. "Protected chords in the web: secure P2P framework for decentralized online social networks". In: *2015 IEEE international conference on peer-to-peer computing (P2P)*. IEEE. 2015, pp. 1–5 (Page: 102).

[180] Kalman Graffi and Newton Masinde. "LibreSocial: A peer-to-peer framework for online social networks". In: *Concurrency and Computation: Practice and Experience* 33.8 (2021), e6150 (Page: 102).

[181] Dorothy Neufeld, Joyce Ma, and VisualCapitalist. *The History of Innovation Cycle*. 2020. URL: https://www.visualcapitalist.com/the-history-of-innovation-cycles/ (Page: 102).

[182] Danny Ecker. *Altstadtfest (Old Town Festival) celebrating the opening of the rebuilt old town with a drone show*. URL: \url{{https://www.shutterstock.com/image-photo/frankfurt-germany-sep-29-2018-altstadtfest-1192594309}} (Page: 103).

[183] Sam Shead. *Elon Musk says production of Tesla's robot could start next year, but A.I. experts have their doubts*. 20022. URL: https://www.cnbc.com/2022/04/08/elon-musk-says-tesla-is-aiming-to-start-production-on-optimus-next-year.html (Page: 104).

CV

Ahmad Reza Cheraghi

27.11.1981, Tehran/Iran

German

Work Experiences

04.2014 – Present	**PhD Student & Research Assistant at University of Duesseldorf (Heinrich Heine University)** *Natural Inspired Algorithms for Robot Swarms*
06.2019 – Present	**Co-Founder and CEO at HireQu** *A platform for remote Software Developer*
08.2013 – 12.2020	**Owner and Manager of LakeSideFlat** *Sales and Marketing, Cost-Controlling, Booking/Guest-Manager and -Support*
03.2013 – 10.2013	**Abelanalytics, Berlin** *Co-Founder & Prototyp-Developer of a "Wifi-Ping Capturing System based on Raspberry Pi" for analyzing Customers Behaviour in Retailstores*
05.2007 – 06.2012	**SMS-SIEMAG AG, Duesseldorf** *International Commissioning Engineer for the Automation Systems of Steel-Plants (Continues Casters)*

Assignment Abroad as a Commissioning Engineer from SMS-Siemag:

01.2012 – 06.2012	**TATA-Steel Ltd.**, Jamshedpur/India
04.2010 – 09.2011	**Mobarakeh Steel Co.**, Mobarakehl/Iran
11.2009 – 02.2010	**Bhushan Steel Ltd.,** Angul/India
09.2009 – 10.2009	**Thyssen AG,** Bochum/Germany
01.2009 – 06.2009	**JSW Steel Ltd.,** Belary/India
09.2008 – 11.2008	**AmurMetal,** Komsomolsk na Amur/Russia
03.2008 – 06.2008	**Bhushan Steel Ltd.,** Jharsuguda/India
09.2007 – 11.2007	**Severstal Ltd.,** Columbus-Mississippi/USA

07.2007 – 07.2007 **ArcelorMittal,** Krakau/Polen

Publications

2021:

In Proceedings of the Springer 2021 Intelligent Systems Conference (IntelliSys). Volume 3.:

- **Past, Present, and Future of Swarm Robotics**

2020:

In Proceedings of IEEE 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)).:

- **A Leader Based Coating Algorithm for Simple and Cave Shaped Objects with Robot Swarms.**
- **Opportunistic Network Behavior in a Swarm: Passing Messages to Destination**
- **Robot Swarm Flocking on a 2D Triangular Graph**
- **General coating of arbitrary objects using robot swarms**
- **Phototactic movement of battery-powered and self-charging robot swarms**
- **Universal 2-dimensional terrain marking for autonomous robot swarms**

In Proceedings of the 3rd International Conference of Intelligent Robotic and Control Engineering (IRCE):

- **Prevention of Ant Mills in Pheromone-Based Search Algorithm for Robot Swarms**
- **Swarm-sim: A 2d & 3d simulation core for swarm agents**

2016:

In Proceedings of the 2016 25th International Conference on Computer Communication and Networks (ICCCN):

- **The State of Simulation Tools for P2P Networks on Mobile Ad-Hoc and Opportunistic Networks**

Academical Background

08.2015 – 08.2017 **Duesseldorf Business School**
Master of Business Administration

04.2006 – 02.2007 **University Dortmund**
Doctoral researcher

10.2001 – 10.2005 **University of Applied Sciences Niederrhein**
Dipl. Ing.(FH) Computer Engineering

Abroad University

09.2003 – 01.2004 **Dundalk Institute of Technology, Dundalk/Ireland**
Web development, Project management and Data networking

School Background

03.2001 – 07.2001 **College Niederrhein, Krefeld/Germany**
Certificate of access to German University

09.1996 – 07.2000 **Tehran International School, Tehran/Iran**
International Diploma

08.1992 – 07.1996 **Benzenberg Realschule, Duesseldorf/Germany**

08.1988 – 07.1992 **Elementary School, Duesseldorf/Germany**

Further Education

10.2012 – 02.2013 **Heinrich Heine University, Duesseldorf**
iOS-Programming & Unified Messaging

Volunteering

03.2016 – Now **Chairman of the Board**
Düsseldorf Business School Alumni e.V.

05.2015 – 05-2019 **Doctoral Representative**
i-GRAD, Heinrich Heine University

Skills & Languages

Programing languages: Python, C, C++, Objective C & Assembler
PHP, Javascript, Bash & Perl
Database: Oracle, PostgreSQL & MySQL
APIs/Frameworks: OpenGL, React, React Natice, NodeJS
IDEs: MS Visual Studio, Eclipse, PyCharm
Markup Languages: HTML, CSS, JSON & XML
OS: MS Windows, Linux & Apple OS X
Office-Products: MS Office & Open Office
Networking: CCNA 1 & 2
Languages: German, English & Farsi

Hobbies & Sports

Hobbies : Cooking, travelling, singing, playing guitar, reading, researching, and philosophizing

Sports : Dancing (Salsa & Bachata) , ice skating, rollerblading, jogging, body building, and meditating

Ahmad Reza Cheraghi, May the 4th 2022