UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Collective Adaptation and Symmetry Breaking to Environmental Changes in a Swarm of Kilobots Monitoring a Dynamic Environment

Kollektive Anpassung und Symmetriebrechung an Umweltveränderungen in einem Schwarm von Kilobots, die eine dynamische Umgebung überwachen

Masterarbeit

verfasst am
Artificial Intelligence Research Laboratory of the Université Libre de Bruxelles (IRIDIA)

im Rahmen des Studiengangs
Robotics and Autonomous Systems
der Universität zu Lübeck

vorgelegt von
Till Aust

ausgegeben und betreut von
Prof. Dr. Heiko Hamann

mit Unterstützung von
Dr. Andreagiovanni Reina

Lübeck, den 15. Juni 2022

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Till Aust

Zusammenfassung

In dieser Arbeit befassen wir uns mit einer kollektiven Überwachungsaufgabe, bei der ein Roboterschwarm eine gemeinsame Entscheidung über das vorherrschende Merkmal in der Umgebung treffen muss. Während sich frühere Arbeiten auf Szenarien konzentrierten, in denen der Schwarm in einer statischen Umgebung agieren musste, in der ein Merkmal dem anderen eindeutig überlegen war, betrachten wir hier anspruchsvollere und realistischere Szenarien. Das erste Szenario ist dynamisch, d.h. Merkmale der Umgebung verändern sich im Laufe der Zeit, was den Roboterschwarm dazu zwingt, seine Meinung zu überdenken und seine Entscheidungen zu ändern. Im zweiten Szenario, in welchem die Merkamle nicht von einander zu unterscheiden sind (gleich stark), muss der Schwarm in der Lage sein die Symmetrie zu brechen und eine Alternative zu wählen. Zuletzt betrachten wir ein drittes Szenario, in dem beide Aspekte kombiniert werden und der Schwarm sich sowohl an Umweltveränderungen anpassen als auch die Symmetrie brechen muss. Wir implementieren ein minimalistisches Roboterverhalten, das aus reaktiven Regeln besteht, um Umweltinformationen zu verarbeiten und einfache Abstimmungsnachrichten zwischen Robotern in Kommunikationsreichweite auszutauschen. Um das Verhalten zu testen, erweitern wir einen bestehenden populären Schwarmrobotik-Simulator. Dabei haben wir ein Plugin entwickelt, welches die erweiterte Realitäts Umgebung Kilogrid simuliert. Wir führen eine große Anzahl von physikbasierten Simulationen durch. Im ersten Szenario stellen wir fest, dass der Roboterschwarm nur in der Lage ist, sich mit lokalen Kommunikationsreichweiten anzupassen, was in der Literatur als *weniger ist mehr*-Effekt bezeichnet wird. Darüber hinaus beobachten wir einen neuen, kontraintuitiven Effekt, den *langsamer ist schneller*-Effekt, durch den sich der Roboterschwarm schneller an eine Umweltveränderung anpassen kann, wenn die Roboter die Umwelt langsamer messen. Im zweiten Szenario stellen wir fest, dass der Schwarm nur bei großen Kommunikationsreichweiten in der Lage ist, die Symmetrie zu brechen. Diese Ergebnisse führen zu dem Problem, dass es keine statische Kommunikationsreichweite gibt, die es dem Schwarm ermöglicht, sich sowohl an Veränderungen anzupassen als auch die Symmetrie zu brechen. Daher entwerfen wir ein neues Roboterverhalten, welches den Robotern erlaubt, ihre Kommunikationsreichweite individuell zu wählen, basierend auf ihren lokalen Informationen. Wir zeigen, dass unser neues minimalistisches Verhalten die Beschränkungen von statischen Kommunikationsreichweiten überwindet und den Schwarm in die Lage versetzt, in komplexen und realistischeren Szenarien zu operieren. Die Anpassungsfähigkeit an Veränderungen und die Vermeidung von Entscheidungsblockaden durch Symmetriebrechung sind Schlüsselfähigkeiten für den Einsatz von minimalistischen Roboterschwärmen in der realen Welt.

Ein Teil der Ergebnisse dieser Arbeit wird auf der internationalen Konferenz über Schwarmintelligenz ANTS 2022 veröffentlicht, die vom 2. bis 4. November 2022 in Malaga stattfinden wird.

Abstract

In this thesis, we consider a collective monitoring task in which a robot swarm needs to collectively reach a consensus on the predominant feature in the environment. While previous work focused on scenarios where the robot swarm needed to operate in a static environment with one feature clearly superior to the other, here we consider more challenging and realistic scenarios. We consider a first scenario where the environment is dynamic—i.e., the features of the environment can change over time—requiring the robot swarm to reconsider its opinion and change its consensus. We also consider a second scenario where the features are indistinguishable from one another—i.e., they are equally predominant, and the swarm must be able to break the symmetry and select any of the alternatives. Finally, we consider a third scenario that combines both two aspects and the swarm needs to both adapt to environmental changes and break the symmetry. We propose a minimalistic robot behaviour composed of reactive rules to process environmental information and exchange simple voting messages between robots within communication range.

In order to test the proposed behaviour, we extend an existing popular swarm robotics simulator to support the selected experimental platform—i.e., we release a new plugin to simulate a Kilobot swarm operating in the augmented environment Kilogrid—and we run an extensive number of physics-based simulations. In the first scenario, we find that the robot swarm is only able to adapt for local communication ranges reproducing what in the literature is called the *less is more* effect. Further, we observe a new counter-intuitive effect, the *slower is faster* effect, by which the robot swarm is able to adapt quicker to an environmental change when its constituent units—the robots—are slower in sampling the environment. In the second scenario, we find that the swarm is only able to break the symmetry for large communication ranges. These findings lead to the problem that there is no single communication range that allows the robot swarm both to adapt to changes and break the symmetry. Therefore, we propose a new robot behaviour that allows the robots to choose their communication ranges individually, based on their local information. We show that our new minimalistic behaviour overcomes the limitations of static small or static large communication ranges, and makes the swarms capable of operating in complex and more realistic scenarios.

Adaptability to changes and avoiding decision deadlocks through symmetry breaking are key capabilities to the deployment of minimalistic robot swarms in the real-world. This thesis moves the swarm robotics literature one step closer to that day.

Part of the results of this thesis will be published in the international conference on swarm intelligence ANTS 2022 which will take place in Malaga from the 2rd to the 4th of November 2022.

Contents

1

Introduction

Using a swarm of minimalistic robots for solving a monitoring task can be beneficial in certain scenarios, where the environment constraints the individual robot's capabilities (Hamann, 2018; Yang et al., 2018; Torney, Neufeld, and Couzin, 2009). In environments such as the ocean floor or in-body blood vessels (Yasa et al., 2020), it is desirable to have biodegradable devices as we do not want to pollute the ocean nor the human body. In search and rescue missions monitoring the environment plays an important role. These missions are mostly in hazardous environments with high risk of damaging the robots. Therefore making the robots disposable saves budget (Jafferis et al., 2019).
Often the robot swarm cannot be controlled or supervised in the given application scenarios. Further, especially in hazardous environments, single point of failures, e.g., central control, significantly jeopardises the mission. For this reasons controlling the robots via minimalistic decentralised behaviours can be a viable option. The minimalistic requirements allow implementation on simpler platforms, such as nano- and microrobots (Gauci et al., 2014; Özdemir et al., 2018).
We study the task of collective monitoring of a dynamic environment. This task requires the swarm to reach a consensus on the current state of the environment, therefore requires the swarm to make a collective decision. In particular, we investigate the best-of-n problem where the swarm has to select the best option among n alternatives. In our study the options are represented by different colours and the options' qualities are determined by the colour concentrations in the environment. The task of the robot swarm is to reach a consensus in favour of the colour with the highest concentration, i.e., the option with the highest quality. We consider two different scenarios.
While monitoring an environment, it can happen that the qualities of the available options change, e.g., the gas concentrations change due to a broken pipe. Therefore, the first scenario is adaptive best-of-n in which we assume a dynamic environment with two different colours, which change in concentration over time. This scenario allows us to study the robot swarm's ability of adaptation. We conduct the experiments of the adaptive best-of-n scenario to reproduce the *less is more* effect observed by Talamali et al. (2021), by which less intra-robot communication lead to a more adaptive swarm. Thereby, we also observed the *slower is faster* effect which has not been documented before in robot swarms making collective decisions, and indicated that slower updates by the individual robots lead to faster collective adaptation to environmental changes (Aust et al., 2022).

During monitoring an environment it can happen that one or more options have similar qualities. That is why we study the robot swarm's ability of symmetry breaking in a second scenario. The task of the robot swarm in this scenario is to collectively decide on one of the n options, which have equal quality. We find that the current robot behaviour is able to break the symmetry for large communication ranges, where it is not able to adapt due to the *less is more* effect.
Therefore, we propose a new extension to the robot controller in which the robots can autonomously change their communication range, based on their current (local) situation. We test the new extension on the two considered scenarios, adaptive best-of-n and symmetry breaking. Further, we introduce a third scenario: adaptive symmetry breaking, where we combine the two scenarios. We show that the extension of dynamically choosing the communication range works in all three scenarios and is a promising solution to overcome the limitations of static local or static global communication ranges.
We implement the environment using the Kilogrid, a unique augmented reality for the Kilobot platform, later shown in Fig. 4.2. Besides allowing to implement arbitrary best-of-n-problems, we use the communication infrastructure of the Kilogrid to enhance the robot's communication capabilities. For the minimalistic robot behaviour, executed on the Kilobot platform, we extend the controller of Talamali et al. (2021) by adapting it to the new environment and improving the robot's behaviour, as further described in Section 4.3.
The experiments are performed using the simulator ARGoS 3 (Pinciroli et al., 2012), which we extended by implementing and releasing the source code of a plug-in to simulate the Kilogrid platform[1].

Structure of the Thesis

In this section, we give an outline of the thesis structure. In Chapter 2, we present the fundamentals for this thesis. Thereby, we first give an introduction to swarm robotics and collective decision-making, followed by presenting the best-of-n problem, the task of collective perception, the *slower is faster* effect and the *less is more* effect, in Section 2.1. Second, we present the related work in Section 2.2.
In Chapter 3, we present the used robotic platform, the Kilobot robot (Section 3.1) and its augmented reality arena, the Kilogrid (Section 3.2). In Section 3.3, we explain the simulation environment ARGoS 3 and the newly developed extension to simulate the Kilogrid.
In Chapter 4, we present the two scenarios, adaptive best-of-n and symmetry breaking, in Section 4.1, and the robot behaviour in Section 4.2.
In Chapter 5, we present the conducted experiments. Thereby, we define the used metrics in Section 5.1. The experimental set-up, results and discussion for the adaptive best-of-n scenario and the symmetry breaking scenario are presented in Section 5.2 and Section 5.3 respectively.
In Chapter 6, we present an extension of dynamically adjusting the communication range to solve the new problem scenario of adaptive symmetry breaking, which combines the previous scenarios. Therefore, we describe the implementation of the new extension in Section 6.1 and the conducted experiments and results in Section 6.2.

[1] https://github.com/tilly111/adaptive_symmetry_breaking

In Chapter 7, we summarise our study and give an outlook on future work.

2

State of the Art

In this chapter, we present an overview of the literature and explain fundamental concepts used in later work.
We start by giving a short introduction into swarm robotics. Next, we present the fundamentals of collective decision-making, followed by a later classification. Thereafter, we introduce the best-of-*n* problem and the task of collective perception. Afterwards, we describe the *slower is faster* effect and *less is more* effect, which we find in our experiments. With the described fundamentals we can then classify and distinguish our work from other approaches found in literature.

2.1 Terms and Definitions

Swarm Robotics

Dorigo and Şahin (2004) defined, "Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment."
The aforementioned definition imposes certain requirements for the robots. First, the robots need to have some kind of local sensing for interacting with the environment. Second, the robots need to have some kind of communication capabilities, for collaboration and cooperation between the swarm members. Local communication is a key feature of the swarm, as communication is fundamental for local interactions and thus collaboration or cooperation. Thereby, we want to point out that collaboration of swarm members means exceeding mere parallelisation (Hamann, 2018).
From a second definition by Beni (2005), "a group of non-intelligent robots forming, as a group, an intelligent robot. In other words, a group of 'machines' capable of forming 'ordered' material patterns 'unpredictably'.", we can extract further properties, that are, decentralised control, lack of synchronicity, simple (quasi) identical members or quasi-homogeneous members which are mass produced (Hamann, 2018).

Collective Decision-making

An important objective in swarm robotics is the ability for the swarm to make independent decisions. This enables the swarm to behave in an autonomous way, creating the possibility of behaving intelligently. Therefore, collective decision-making is a fundamental capability of the swarm (Hamann, 2018). For autonomous behaviour of the robot swarm, each swarm member should behave autonomously, i.e., each individual robot should do independent decisions. These independent decisions should contribute to the collective decisions of the robot swarm.
Decision-making describes the process of choosing one action from different alternatives, i.e., the robot decides which action to take. Individual decisions are often made under uncertainty, because the robots are only capable of local sensing and local interaction. In swarm robotics, it becomes more difficult, as a group of robots needs to make a decision, that is why, collective decision-making is more complex and might show effects of social influence and, for example, group polarization (Hamann, 2018).
In collective consensus decision-making it is tried to avoid "winners" and "losers" of the decision. While the majority approves a decision, the minority goes along, with the right to veto at any time. The procedure of collective consensus decision-making is complicated to implement. A more common approach are voting-based methods. Most of the voting-based methods are extending the classical voter model, first presented by Holley and Liggett (1975). The classical voter model can be seen as a connected graph. Each node is a voter, which can interact with other voters it is connected to, i.e., its neighbours. At any given time the voter's opinion can be either 0 or 1. The update of opinion happens, at random times, to one randomly selected voter. Then the chosen voter (randomly) selects one neighbouring voter and adopts its opinion. For making the correct decision, i.e., deciding towards the best option for the swarm, the classical voter model can be extended to the weighted voter model, which means that the quality of the option influences the dissemination of that option (Valentini, Hamann, and Dorigo, 2014).
In this study, we implement an extension of the weighted voter model following the social interaction pattern cross-inhibition, which is motivated by honeybee nest-site selection (Seeley et al., 2012). This model is formalised (Reina et al., 2015; Reina et al., 2017b) and a minimalistic approach further described in Section 4.2. We chose cross-inhibition as it leads to more stability in the process of finding consensus and enables the voters to break decision deadlocks in case of equal-best options (Seeley et al., 2012).

For swarm robotics Brambilla et al. (2013) formalised two categories of collective decision-making problems:

1. the problem of consensus achievement and
2. the problem of task allocation.

In the first category, a swarm of robots tries to make a common decision on a certain matter. The second category describes how agents allocate themselves to tasks, in order to maximize the performance of the swarm.
In this work, we study the first category, which is further classified by Valentini, Ferrante, and Dorigo (2017). They subdivide the category of consensus achievement into the

discrete and continuous case. While the continuous case encompasses problems with infinite and measurable choices, e.g., selection of a common direction of motion by a swarm implementing a flocking behaviour, the choices of the discrete problem are finite and countable. We study the category of discrete consensus achievement problems of decision-making.

Best-of-n Problem

The best-of-n problem is an instance of the discrete consensus achievement category.
A swarm of N robots found a solution for a specific best-of-n problem, if it makes a collective consensus decision on one out of n available options. The robot swarm made a collective consensus decision if a *large majority* $M \geq (1 - \delta)N$ is in favour of the same option, e.g., $0 \leq \delta \ll 0.5$ (Valentini, Ferrante, and Dorigo, 2017). Each option i has a quality ρ_i. The robots can make noisy estimates of the qualities. The estimates influence the decision process in the weighted voter model, allowing us to say that the robot swarm found the *right* solution to a specific best-of-n problem, if the swarm makes a collective consensus decision on one option with the highest quality.
We can further categorise the best-of-n problem by differentiating on the options qualities, they can be either symmetric (all options have the same quality, $\rho_i = \rho_j \forall i, j \in \{1, ..., n\}$) or asymmetric (the options have different qualities, $\rho_i < \rho_j \forall i, j \in \{1, ..., n\}$).

Collective Perception

Collective perception is a special case of the consensus achievement task in collective decision making (Valentini et al., 2016b).
The robot swarm's task is to solve a best-of-n problem, that is to identify a predominant feature, e.g., concentration of a colour, scattered in an unknown environment. The environment is considered to be much larger than the individual robot, thus a single robot can only make local assumptions by estimations based on its current location (local perception). However, this local information is generally not enough to draw reliable conclusions for the whole environment. Therefore, it is beneficial to have multiple agents working together to cover larger areas and collect more spatially distributed information. The individually gathered information is then combined through local robot interaction for making collective decisions.

Slower is Faster Effect

The *slower is faster* effect is a phenomenon which can be observed in many real life situations.
In ecology, we can observe the *slower is faster* effect, e.g., in the predator prey relationship (Slobodkin, 1961). If predators consume their prey too fast the prey population will decline, meaning no prey to consume, leading to a declining predator population. Thus, "prudent" predators will spread faster than greedy predators because they let the prey population recuperate.
The *slower is faster* effect can also be observed in social dynamics. In the voter model

the agents may change opinion depending on the option of their neighbours to eventually converge to a state where each agent has the same opinion. If the agents change their opinion too fast, it might delay convergence (Stark, Tessone, and Schweitzer, 2008a; Stark, Tessone, and Schweitzer, 2008b). Thus, faster (individual) opinion switches will not lead to faster convergence of the agent swarm.
A third example of the *slower is faster* effect can be found in harbour logistics using automated vehicles for container transport. By reducing the speed of the individual vehicles, the required safety distances between vehicles can be lowered, so that there were less conflicts of movement, resulting in that the automatic guided vehicles had to wait less. Thus, the overall transportation time could be reduced by increasing the individual movement time (Gershenson and Helbing, 2015).
All these scenarios have in common that they can be modelled as complex dynamical systems composed of many non-linearly interacting agents (Gershenson and Helbing, 2015). To summarize, the *slower is faster* effect describes that some system is able to accomplish its task faster if the individual components, i.e., agents or robots, perform their task slower.

Less is More Effect

The *less is more* effect was first observed in robot swarms by Talamali et al. (2021). They found that by limiting the communication capabilities of the individual robots, i.e., reduce the robot's individual communication range, the swarm performs better in making a consensus decision in favour of the better option. While by increasing the communication capabilities, i.e., all to all communication between swarm members, the swarm is not longer able to make a consensus decision in favour of the better option. A requirement for the *less is more* effect is that recruitment takes time, i.e., there need to be a delay between the moment a robot is recruited and the moment the robot recruits other robots. In the study of Talamali et al. (2021) the delay was implemented in the following way: after recruitment the robots first needed to make an estimate of the option they have been recruited to, before recruiting other robots to their option.
The counter-intuitive *less is more* effect can be explained via the social impact of committed subpopulations of unbalanced sizes. A large majority is able to repeatedly mute minorities that make temporary discoveries of alternative options. The minority's opinion is slow to gain traction in the population as new recruits are slow in becoming vocal and are quickly reverted to the majority's opinion. When the communication range is large, or equivalently when the robot density is high, any minority is in contact with the large majority at all time. Instead, sparse connectivity, due to small communication range or a low robot density, reduces the importance of subpopulation sizes. Interactions are sporadic (often limited to pairs) and the collective dynamics is governed by opinion quality (in our case encoded via messaging frequency).

2.2 Related Work

Here we give an overview of the related work that investigated collective perception of an environmental feature in swarm robotics and how it differs from our approach.
Similar to the work of Talamali et al. (2021), we opt for a minimalistic approach, to run our algorithms on robots, which have minimal requirements in terms of memory, computation, sensing and communication capabilities.
Shan and Mostaghim (2020) proposed the collective decision-making strategy Distributed Bayesian Hypothesis Testing (DBHT) to solve the collective perception problem. In their method, the robots first individually form their estimation of likelihood of various hypothesis by observing their immediate surrounding environment. Then a leader periodically collects opinions from other robots, and forms the final estimate of the whole swarm. The DBHT approach is extended by Shan and Mostaghim (2021)'s work. They introduce the Distributed Bayesian Belief Sharing (DBBS), which removes the central leader, needed in DBHT, for forming the swarm's final estimate. However, in both approaches, DBHT and DBBS, the robots are required to store all available alternatives and all received messages, whereas in our work the robot only stores a single opinion, i.e., the colour it considers predominant and the estimated colour concentration, the last message received from a neighbour, and a temporary variable to estimate possible environmental changes.
Ebert, Gauci, and Nagpal (2018) studied the multi-feature collective decision making problem. The robots needed to decide on multiple features simultaneously. Similar to Valentini et al. (2016b), the robot controller is separated into a measuring phase and a dissemination phase. Thereby, the robot keeps a belief, concentration and decision for every feature in the environment. In Ebert et al. (2020) each robot employs a Bayesian model of the fill ratio and makes a decision using credible intervals of the posterior distribution. All of these approaches use computation based on Bayesian inference. Bartashevich and Mostaghim (2021) studied the collective perception problem using operators from epistemic logic. Implementing the robot behaviour based on Bayesian inference or epistemic logic makes the robot controller more complex and thus more computational demanding compared to our robot behaviour, that is defined by a small finite state machine with standard reactive transitions.
Another limitation of our robots is the sensing capability. The implementations of Shan and Mostaghim (2020), Shan and Mostaghim (2021), and Ebert et al. (2020) require the robots to be able to determine the predominant element at every measurement. In our approach the robot is only able to measure the presence of absence of a certain element, i.e., the element the robot is currently measuring.
Previous work needed rich inter-robot communication to maintain required shared collective knowledge. In the case of DBHT, a centralised entity is required. In our approach the robots only send simple messages containing few bits, indicating their preferred element. In the case of $n = 2$ this can be reduced to one bit of information.
Related work in the field of collective perception, that is comparable in its simplicity of individual robot requirements is that of Valentini et al. (2016b). So far all studies assumed a static environment for the collective perception problem. Only few studies considered dynamic environments. The work of Prasetyo et al. (2018) and Prasetyo, De Masi, and Fer-

rante (2019) considered a dynamic environment for the site selection scenario with $n = 2$ options. Similar work, which also considers a site section scenario was done by Soorati et al. (2019) and Talamali et al. (2021). We consider a dynamic environment for the collective perception scenario.

3

Material and Methods

This chapter is split in two main parts. The first part presents the needed hardware. This includes the robotic platform Kilobot, described in Section 3.1 and its unique augmented reality arena Kilogrid, described in Section 3.2. The other part covers the used and developed software of this work. Thereby, we begin by introducing the simulation environment ARGoS 3 in Section 3.3. Finally, in Section 3.3, based on the simulation environment and the real hardware, we developed a plug-in for ARGoS 3 to simulate Kilogrid.

3.1 Kilobot

For the conducted experiments we chose the Kilobot robot which is widely used in swarm robotics research (Rubenstein, Cornejo, and Nagpal, 2014), see Fig. 3.1. The low cost and mass programmability of this platform allows for large scale experiments. Even though the cost per robot is low, it comes with all the capabilities needed for running robot swarm experiments, such as locomotion (moving forward and rotating), communication with neighbouring peers (also measure distance), sufficient memory and computing power to run the behaviour controller.
The robot is equipped with two vibrating motors. The slip-stick principle is used to generate a forward motion out of rotation. The two motors allow a differential like behaviour (activate one for rotation; both for driving straight). This allows the Kilobot to move approximately 1 cm/s and rotate approximately 45 degrees/s. The slip-stick motion approach was chosen because of its low cost, but it comes with the problem of precise movement. Due to the slip-stick based motion, there is no real form of odometry, which makes it impossible to be accurate over a longer distances.
The robots do not have capabilities of directly sensing (no sensors are included) besides a visible light sensor, which we do not use for our experiments.
The Kilobot has an infrared transmitter and receiver on the bottom side. It is a wide-angle transmitter with a half power of 60° from the robot's downward pointing vertical axis. This allows for local communication of up to 10 cm (Rubenstein, Ahler, and Nagpal, 2012). Normally, the communication module is used as in Fig. 3.2. The transmitter emits infrared light, which is reflected by the floor of the arena and then can be received by neighbouring robots. Messages are transmitted by pulsing the transmitter according to

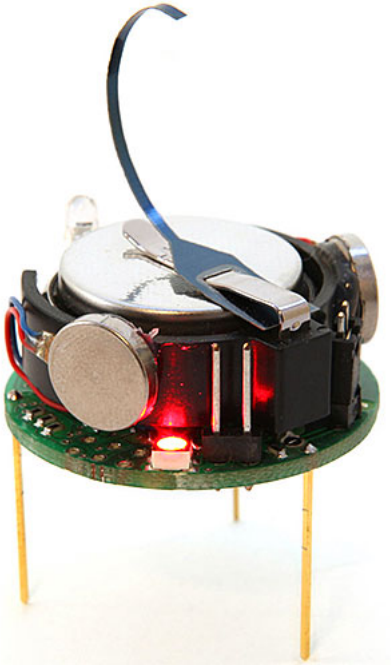


Figure 3.1: This is a Kilobot.

standard line coding technique. This allows a theoretical maximum communication rate of 30 kb/s. However, empirical results show messaging with this high frequency causes a lot of interference (Pinciroli et al., 2018). To avoid these interference, the maximum communication frequency is set to sending infrared messages with 2 Hz. One infrared message consists of 8 byte payload and 1 byte header. There can be still the interference, because all robots use the same infrared channel for communication. To mitigate this problem a standard carrier sense multiple access with collision avoidance (CSMA/CA) has been implemented. However, crowded areas will have a significant drop of channel bandwidth.
The receiver can also measure the intensity of the incoming message and thus determine how far the message travelled. This is not used, because in our approach the robots do not speak directly with each other, as described in detail in Section 3.2.
Another problem in large robot swarm experiments is the limited scalability due to robot programming. The Kilobot platform overcomes this issue by offering to be programmed over air with the overhead controller. The overhead controller is an infrared transmitter, which can be connected to the computer via a serial connection. It then can be used with the open source software KiloGUI[2] for controlling the robots. Besides programming large quantities of robots at the same time (broadcasting), other commands such as running and stopping the robot, checking its battery status, calibrating the Kilobot's motors and ID assignments are possible.
The on-board capabilities are very limited on this robotic platform. In our experiments, as described in Section 3.2, we employed the Kilogrid which allows us to amplify the Kilobot's communication range, as well as to equip the Kilobot with a range of virtual sensors,

[2] https://github.com/acornejo/kilogui

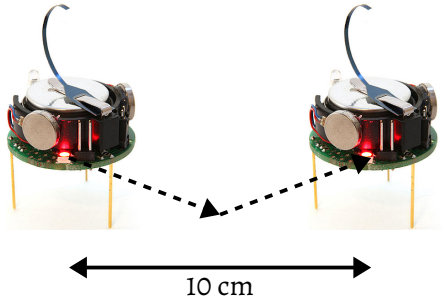


Figure 3.2: Communication between two Kilobots. One robot emits an infrared message, which is reflected by the ground and then received by a neighbouring robot.

such as a virtual GPS or a virtual sensor for reading the ground's colour. This can be generalised to arbitrary sensing and thus allows for a dynamic and fast experiment development.

3.2 Kilogrid

The Kilogrid is a worldwide unique augmented reality Kilobot arena developed by Antoun et al. (2016). It is open source[3] to enhance the reproducibility for swarm experiments. In the following we give a short overview of its structure, followed by an exemplification of how we utilise these structures to conduct our experiments.
The Kilogrid, we are using, is a 1000 mm by 2000 mm large grid of modules and a dispatcher. Each module is 100 mm by 100 mm. The dispatcher allows to interface the modules with a computer, allowing to program each module individually. On top of the modules is a single 8 mm thick transparent Plexiglas surface on which the Kilobots can move. One module consists of four cells, which are 50 mm by 50 mm, arranged as in Fig. 3.3. Further, each cell has one infrared transceiver and two RGB LEDs driven in parallel, pointing upwards. The infrared transceiver is used to send and receive infrared messages from the Kilobot through the Plexiglas surface. The RGB LEDs can be used to indicate the internal state of the cell.
Moreover, the dispatcher incorporates all the functionalities as the original overhead controller. The dispatcher can be controlled, similar to the overhead controller, via a serial connection to a computer. Therefore, the KiloGUI has been extended to allow, in addition to the Kilobot commands, the user to program the modules and set-up experiments. This allows easy programming and calibrating of the Kilobots and Kilogrid modules.
The communication infrastructure between modules and dispatcher uses a Controller Area Network (CAN). The programming environment of the Kilogrid has been developed as an extension of the kilolib library[4]. This extensions allows to program the modules in a similar manner as the Kilobot. The module implements a `setup` function which is called in the beginning of the experiment. In the `setup` function a user can pass parameters to the module through a `.kconf` file. The structure of the `.kconf` specifies which modules

[3] kilogrid https://www.giovannireina.com/kilogrid/about.html.
[4] https://github.com/acornejo/kilolib.

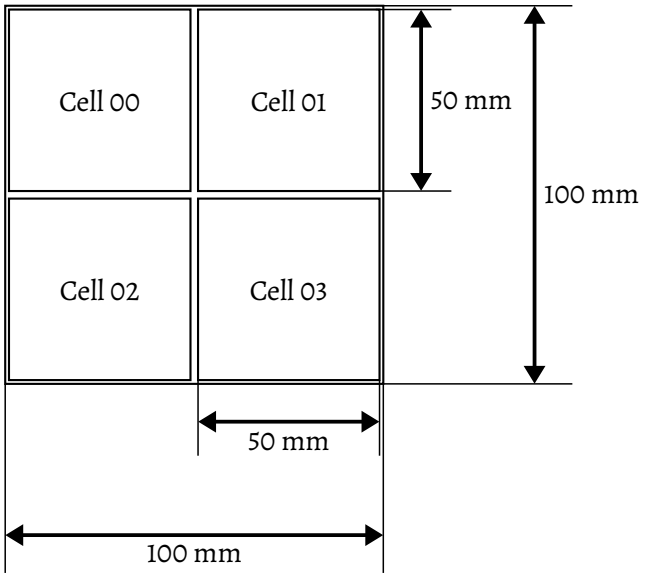


Figure 3.3: Layout of the four cells from one module.

receive what parameters. It is possible to address the modules individually, by row or by column. The module implements a `loop` function, which is called every control cycle. The message reception (infrared and CAN) is implemented as a callback function, similar to the infrared message reception implementation of the Kilobot.
The electronic architecture of a module features an ATmega328P microcontroller unit (MCU) for executing user-defined programs. A CAN interface handles the communication with the dispatcher. The network stack is built up on the standard CAN protocol, which allows module to module communication as well as module to dispatcher communication. The dispatcher is also equipped with an ATmega328P microcontroller. Through fixed positions on the grid (and fixed addresses) the modules are independent communicating entities. The current CAN bus system can only handle up to 112 nodes, thus the dispatcher needs to be adapted in the following way to virtually connect all 200 modules. Its CAN interface is provided with two CAN repeaters that divide the CAN network into four buses, which are physically separated but virtually connected. This allows a maximum number of $4 \times 112 = 448$ nodes. The current network speed is 250 Kbps, for the sake of increasing the maximum possible number of nodes, this would go down. The four infrared transmitters and receivers are multiplexed and independently driven by the MCU. The received infrared signals are processed such as in the Kilobot (i.e., amplified and filtered), permitting to measure the distance between a transmitting Kilobot and a receiving cell. To prevent cross-talk of adjacent cells, IR barriers were introduced.
The high flexibility of the Kilogrid provides many desirable features. First, the Kilogrid makes it easy to design an environment as described in Section 4.1, by assigning one option to each cell. Second, collecting data is very convenient. Finally, the most important argument is that we can overcome the limited communication range of the Kilobot by virtualising the communication.

In the following we describe how we implemented the virtual communication transmis-

sion between Kilobots utilizing the Kilogrid and thus enhancing the communication capabilities of the Kilobot. In Fig. 3.4, we present the novel approach of virtually forwarding

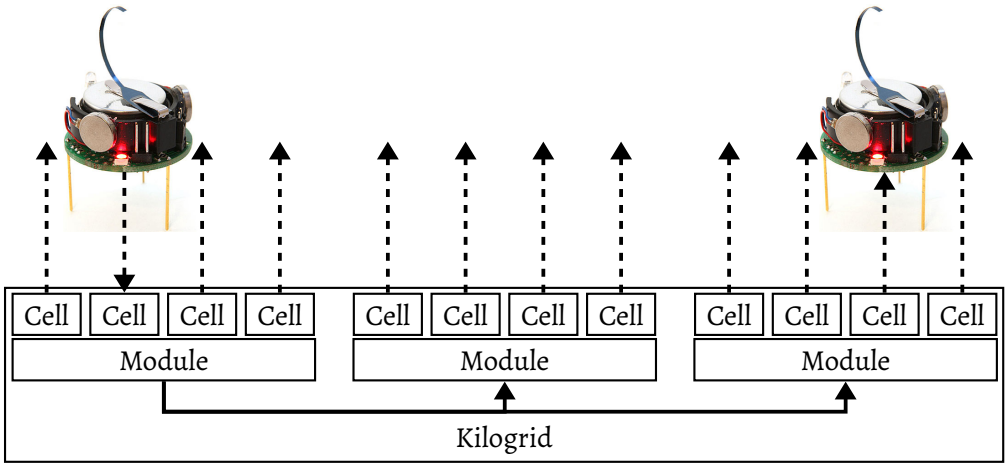


Figure 3.4: Forwarding the Kilobot message with the Kilogrid.

Kilobot messages. First, the sending Kilobot transmits a message containing the range it wants the message to be distributed. This happens with the normal Kilobot message sending protocol, described in Section 3.1, which is supported by the Kilogrid. Next, the message gets processed in the receiving Kilogrid module (the information which cell received the message from the Kilobot remains). The processed message is then transformed to a CAN message. The corresponding CAN message is then send to all other modules via a CAN wide broadcast. At first, this seems counter intuitive, as we would expect that sending the message only to adjacent cells would be a more scalable solution and not all modules have to process, whether they are meant to receive the message.
To understand this design choice we briefly explain how the messaging in this bus system works. One can imagine the bus system as a single channel, where all the nodes listen all the time. For receiving messages, filters need to be set in the corresponding nodes, which only allow certain messages to be passed through to the user level. Because there is only one channel, it is only possible that one node speaks at a time. The given hardware limits the number of filters, such that we can only implement individual messaging and broadcasting.
The drawbacks of individually messaging all the neighbouring cells are that we would need to send multiple CAN messages (up to 200 in the case of global communication) per received infrared message of one robot. This would lead to a huge delay, as there can be only one sending node at a time, hence we would have to sequentially send all the messages. Another point is, that by default all nodes receive all messages, this implies that for better scalability we should try to minimize the number of sent messages.
Next, we explain how the message processing in the receiving module is done. The received message is first processed by a callback function. In this callback function, we first check if there is a cell, which is intended to receive the message by using the euclidean

distance measure,

$$\sqrt{(x_s - x_{my})^2 + (y_s - y_{my})^2} < r, \tag{3.1}$$

with (x_s, y_s) being the position of the sending cell, which initially received the message from the Kilobot, (x_{my}, y_{my}) being the position of the receiving cell and r_c the range the robot wants the message to be send. We decided to use the euclidean distance measure, because it approximates a circle which is the usual form of communication range, but it can be replaced by any user defined distance measure. If the receiving cell is in range, the Eq. (3.1) holds, the information of the message are stored in a temporary storage, which gets read out in the next cycle of the module. This also handles the case if multiple messages arrive at similar time, e.g., in the same control cycle. In this case, only the last message is forwarded, as the temporary storage gets overwritten by the last received CAN message. In the main loop of the module, the received CAN message is then converted to an infrared message and send. If there is a Kilobot on top of the sending cell, it receives the message and can process its information.

Augmented Reality for Kilobots (ARK)

Because of the simplicity of the Kilobot, there is a desire for augmented realities to enhance its capabilities. Besides the Kilogrid, there are other augmented realities, such as the augmented reality for Kilobots (ARK), which is widely adopted (Reina et al., 2017a; Font Llenas et al., 2018; Talamali et al., 2021; Talamali et al., 2020; Minimalist Robot Swarms, 2022; Pratissoli et al., 2019). It has many advantages, such as easy calibration, its low cost and relatively large research community. The disadvantage however, is the very limited communication capability, which is fundamental for our approach of virtualising the communication. As aforementioned, the Kilogrid is able to overcome these limitations, due to its decentralised nature.

3.3 Kilogrid in ARGoS 3

For fast prototyping and testing algorithms we need a simulation environment, which resembles the reality as close as possible. Therefore, we chose to implement the Kilogrid in the physics-based simulator ARGoS 3 by Pinciroli et al. (2012). We chose this simulation environment because of the performance and the customizability. Further, this simulation environment already supports the robotic platform Kilobot as a plug-in (Pinciroli et al., 2018).
The fundamental design concept of ARGoS 3 includes, that the programming of the robots, in our case the Kilobot, should be as close to reality, as possible. Thus, it offers the same interfaces as the real robot does. Inspired from this approach, we also opted to design the *Kilogrid plug-in*, such that it offers the same interfaces as the real Kilogrid.

ARGoS 3

In this section we give a short overview of the structure of ARGoS 3 for a better understanding of the design and integration of the *Kilogrid plug-in*.
From a user perspective there are three important files, as depicted in Fig. 3.5. The first file is the experiment configuration file. It is a XML file with the extension `.argos`. This file defines how the experiment is conducted, e.g., it defines the robot controller, the environment properties and the random seed used. The second file is the robot controller.

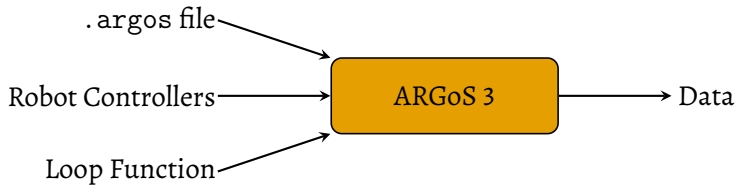


Figure 3.5: Fundamental usage of ARGoS 3. On the left side are the required files the user has to provide to run an ARGoS 3 experiment, which then generates the desired data. Adopted from `https://www.argos-sim.info/user_manual.php`.

It defines the code, which is executed on each robot taking part in the experiment. The code for our experiments is described in Section 4.3. The last file, which is important, is the loop function file. It handles the behaviour of the environment for each time step and thus is the starting point of integrating the Kilogrid into ARGoS 3. In Fig. 3.6 one can see the the fundamental cycle which is executed for any simulation run by the loop function. In the beginning of a simulation run, the `init` function is executed. After-

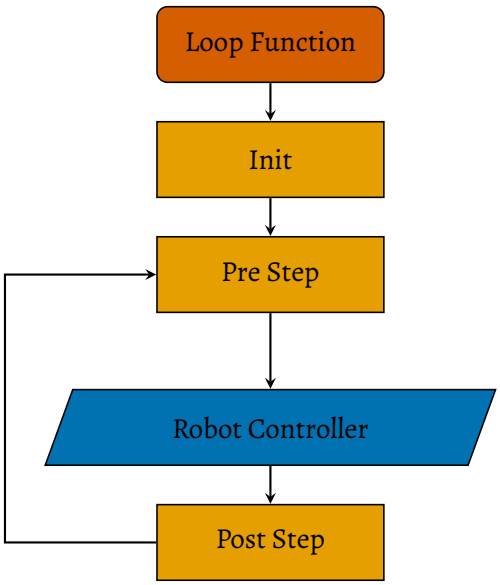


Figure 3.6: Fundamental cycle of the loop function of the ARGoS 3 simulator.

wards, for the specified number of simulation steps, the following loop is executed every simulation step. First, the `Pre Step` function is executed, second, in random order, the

robot controllers are executed for one simulation step and third, the `Post Step` function is executed. In the end the of a simulation run, the `Destroy` function is called.

Kilogrid plug-in

Based on the aforementioned loop, we developed the *Kilogrid plug-in*, which we describe in the following. Thereby, we structure the explanation after the steps taken in the loop.
Init: this function is used to initialise the shared memory of the virtualised modules, which is later used for the communication. It also initialises necessarily needed modules from the simulation environment and the robot tracking. Next, it calls the `.kconf`-parser, which takes a `.kconf`-file and sends its information to the specified (virtual) module. The `.kconf`-file has the same structure as for the real Kilogrid and thus can be deployed one-to-one on the real Kilogrid. Finally, it calls the `setup` function of each (virtual) module, which processes the data, specified by the `.kconf`-file.
Pre Step: in this step, first, the message reception is processed. This can be subdivided into infrared message reception (robot to module) and CAN message reception (module to module).
Beginning with the infrared message reception, we need to utilize the *Debug struct* of the Kilobot plug-in, which is the only shared memory where both the robot controller and the loop function have access, thus the only way to communicate from the robot to the loop function in ARGoS 3. Normally, the *Debug struct* is only used for tracking and other debugging purposes. We extend the current functionality by adding 8 bytes of memory (later used for storing the payload of the message) and a flag, if the robot currently wants to send a message. Next, the loop function cycles through the Kilobots and if the flag is set it copies the message payload to the corresponding module infrared message reception buffer. The corresponding (virtual) module is the (virtual) module which contains the cell beneath the robot and can be determined by the robots position (which is accessible by the loop function). After all the robot messages have been collected and added to the reception buffers, in a next step, the `IR_rx` function of the corresponding (virtual) modules are called. If there are multiple messages in the reception buffer, a random message is chosen following a uniform distribution and all other messages are discarded.
Next, the CAN messages, send by other (virtual) modules, are forwarded. We check for each (virtual) module, if it received a CAN message. This is done by checking the CAN message reception buffer of the (virtual) module. If the buffer is not empty, one message is randomly selected following a uniform distribution, and all other messages received at this time step are discarded. This message is then forwarded to the corresponding `CAN_rx` function of the (virtual) module.
Afterwards, the `loop` function of each (virtual) module is executed. This function implements the logic of the (virtual) module, e.g., further message processing and sending, and is further described in Section 3.3.
Robot Controller: all robot controller get executed for one simulation step in random order.
Post Step: we use the `Post Step` for tracking the robots commitment. Further, we check here if in the beginning of the experiment all robots were initialised correctly and restart the simulation if this was not the case.

Concluding, in these methods we implement the functions of the modules. This includes the message reception of infrared and CAN messages and sending messages according to the module logic.
The (virtual) modules are separated by individual storage. Sending inter-module messages, e.g., CAN messages, is implemented as appending the message to the corresponding message reception buffer.

Kilogrid Module Controller

This section describes the implementation of the Kilogrid module controller, which implements the virtual message transmission described in Section 3.2. We structure this according to the different functions the module needs to implement, namely the `setup`, `loop`, `IR_rx` and `CAN_rx` function.
setup: this function implements the initial set-up of the module before the experiment starts. We specify the x and y coordinates, the role and the option of each cell of the module during this set-up process using the `.kconf` file. The x values range from 0 to 20 and the y values from 0 to 40. There are three different roles a cell can have. The roles help the robot to navigate. The first role is *wall*, which indicates the robot to do a avoidance behaviour and stop sampling, because the robot leaves the environment. The most outer line of cells of the Kilogrid are assinged this role. The second role, assigned to all cells next to the *wall* cells, are the *close to wall* cells. They indicate to the robots that they should trigger the wall avoidance behaviour, but the robots are still allowed to sample. The last role is *normal ground*, where the robots are allowed to sample and follow their random walk. An illustration can be found in Fig. 3.7. Empirical test runs showed, that one line of wall cells

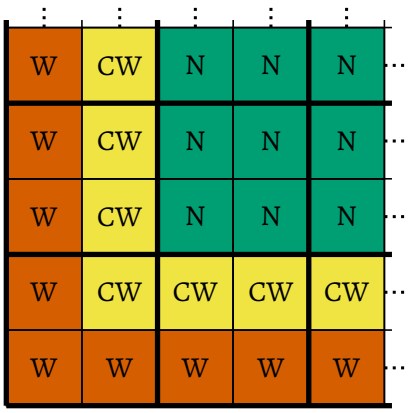


Figure 3.7: Illustration of the role distribution of the cells of the Kilogrid at a corner. The thicker lines indicate the modules, while the thinner lines separate the cells. (W) role wall, (CW) role close to wall and (N) role normal.

leads to many robots getting stuck at the rand. Introducing a second line of wall would take away $38 + 38 + 16 + 16 = 108$ cells. Thus, we introduced the role of close to the wall to maximize the area for the experiment, which allows for more precise differentiation of the element concentration while keeping most of the robots inside the environment.

This leaves us with the following distribution: there are in total 800 cells, from which 116 cells are *wall* cells. The other 684 cells are assigned an option, according to Section 5.1 or Section 6.2 respectively.

loop: this function is called every cycle and contains the logic of a module. The loop function iterates through all the cells sequentially. In the first iteration the module sends, on all its cells, the initial message. It contains initial parameters and the current position, so that the robots can initialise themselves. The contents of the message can be found in Tab. 3.8 (agent message).

The real Kilogrid implements the receiving functions as callback functions, meaning they can be called at any time. To be consistent with the processing and message forwarding, the received messages are first stored temporarily. At the beginning of each control cycle, the latest temporary stored message overwrites the variables used for calculation.

In the following control cycles, the module first checks if it received any infrared message from a robot. If there is a infrared message, the module forwards this message as a CAN message to other modules. Therefore, the contents of the infrared message are copied to CAN message, see Tab. 3.8 (CAN message), then the message is send. Because of the underlying network infrastructure, as described in Section 3.2, we broadcast CAN messages, i.e., send the message once to all modules.

Next, the module processes any received CAN message. This is done by checking if for any cell of the module Eq. (3.1) holds. For the cells that should receive the message, the received CAN message is transformed into a infrared message, by copying the content, see Tab. 3.8 (infrared message).

In the final step, the module sets the infrared message for all cells. The infrared message can either be a status message, or if a CAN message was received, the forwarded message from another robot. The status message is for navigation and sampling of the robot, it contains the coordinates and the option of the ground, see Tab. 3.8 (status message). The forwarded CAN message contains besides the payload, the commitment of the sending robot, the coordinates of the sending cell and the unique ID of the sending robot, see Tab. 3.8 (virtual agent message). These information are used to ensure, that the message is perceived by the right robot and that if the robot is in the middle between two cells, it receives the message only once. How the filtering is done on the robot side can be found in Section 4.3.

IR_rx: this function implements the callback for infrared message reception. These infrared messages are send by robots above the given cell and are the only way for the robots to send messages to the Kilogrid. Thus, this method was previously only used to obtain tracking data from the robots and forward it to the dispatcher, which relays them to the user computer.

We extended this behaviour, by introducing a new message type, the *virtual robot message*. This allows for receiving messages by the robots to virtualise the communication while being able to collect tracking data. The received message is stored in a temporary storage, as mentioned above. Only the last received infrared message is stored in the temporary storage. Each cell has its own temporary storage.

CAN_rx: this function implements the callback for CAN message reception. Previously the CAN bus was used to connect the Dispatcher with the individual modules, to send control commands and collect tracking data. The processing of the control commands is

done in a deeper layer of the module control software. It filters out all control commands and only forwards messages with an unknown message type. This makes it easy to extend the control software of the module for our purposes by introducing the new message type *CAN user message*. For any received *CAN user message* we check, based on the information given by the CAN message, if the module is in range of receiving. For this Eq. (3.1) needs to hold. In case the equation holds, the message is stored in temporary storage. Each module can store only one CAN message temporarily. This is sufficient because in contrast to the infrared messaging, where we can address individual cells, CAN allows only for addressing individual modules.
In essence, we use the module controller and network infrastructure to forward robot messages (extending their communication range capability) and tracking.

Table 3.8: Overview of all message types and their content.

Message Type	Sender	Receiver	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
initial message	module	robot	x (cell)	y (cell)	initial commitment	initial commitment quality	number of options	option (cell)	parameter 0	parameter1
agent message	robot	module	x (robot)	y (robot)	commitment (robot)	communication range	message number	unique ID	unique ID	
CAN message	module	module	message Type	x (robot)	y (robot)	communication range (robot)	commitment (robot)	unique ID	unique ID	
status message	module	robot	x (cell)	y (cell)	option	role (cell)				
virtual agent message	module	robot	x (cell)	y (robot)	commitment (sending robot)	unique ID	unique ID			

4

Problem Description and Robot Behaviour

In this chapter, we formally define our problem in Section 4.1, followed by a description of the implemented environment. In Section 4.2, we present our developed robot behaviour, which aims to solve the aforementioned problem in the given environment.

4.1 Problem Description

In this section, we begin by defining the problems the robot swarm needs to solve. We do so by defining the task of the robot swarm, followed by the definition of the environment. Finally, we explain why the Kilogrid is a sufficient platform for implementing our environment and our problem.

Problem

We consider a collective perception problem, that is, a swarm of robots should collectively decide on the best element they find in their environment. Such elements could be, e.g., different gases or minerals in a remote location and the robot swarm has to autonomously agree on which gas or mineral has the highest concentration. In this study, each element is a colour and the robot swarm has to collectively decide on the most frequent colour present in the environment, see Fig. 4.2.
As indicated in Section 2.2, several studies assumed a static environment, however, this assumption does not meet the characteristics of most real-world scenarios, thus in our study we focus on dynamic environments.
Our study investigates two different scenarios of the collective perception problem: adaptive best-of-n and symmetry breaking.

Adaptive Best-of-n Scenario

In the first scenario, adaptive best-of-n, there is one predominant (best) colour at any point in time. The predominant colour can change over time, because we are investigating dynamic environments. For our robot swarm that translates into the necessity of being adaptive in its process of finding the most frequent colour.

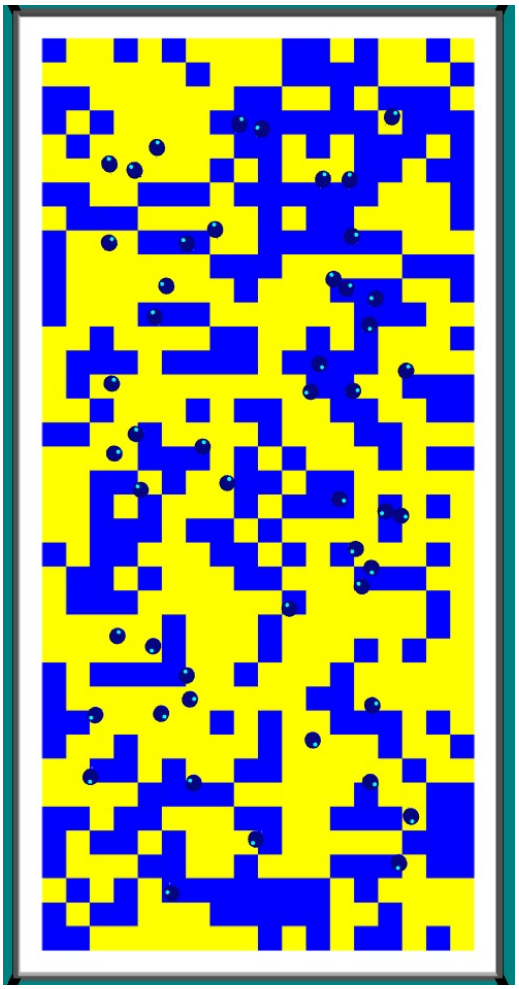
Figure 4.1: Colour concentration during the adaptation experiments. To study the ability of adaptation of the robot swarm, we assume that in the beginning of the experiments ($t = 0$ minutes) the colour concentration instantaneously changes from higher concentration of blue to higher concentration of yellow. One experiment takes 40 minutes.

Adaptation to time-varying environments was studied in previous work by Talamali et al. (2021), who found the *less is more* effect, that is, the robot swarm is more able to adapt to an environmental change when the robots' communication capabilities are limited, e.g., the robots have a small communication range. Similar to Talamali et al. (2021), our experimental scenario is an instance of the best-of-n-problem. We choose $n = 2$ colours, with colour yellow having a higher concentration than colour blue. The task of the robot swarm is to make a consensus decision in favour of the most frequent colour, that is, the swarm makes the correct decision when a large majority of the robot swarm is committed to the colour yellow.
Without loss of generality (w.l.o.g.), we assign blue to option o_1 and yellow to option o_2. The concentration defines the option's quality ρ_i. Without loss of generality in the case of adaptation we have $\rho_b < \rho_y$, where ρ_b and ρ_y are the concentrations of the blue and yellow colour respectively. A way of modelling the problem difficulty is to vary the concentration of the two colours.
We want to test the the ability of the robot swarm to adapt to sudden environmental changes. Therefore, we assume a change of colour concentration in the environment right in the beginning of the experiment, that is we initialise the environment with a higher concentration of yellow than blue, see Fig. 4.1. As blue was the colour with the highest concentration until $t = 0$, the robot swarm starts fully committed to blue. This situation allows us to study, if the robot swarm is able to adapt to the colour with the highest concentration.

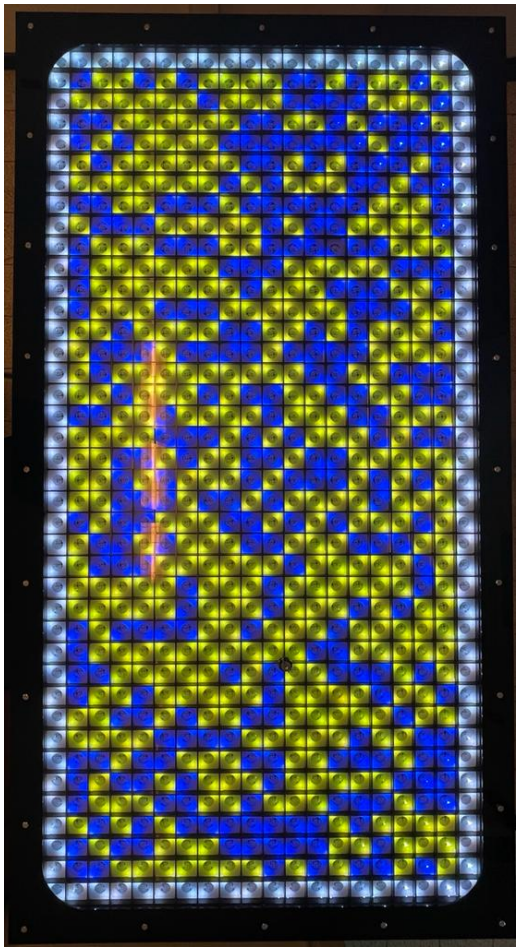
Symmetry Breaking Scenario

Another condition, that can be found in real-world scenarios, is that there are equally good or close to equally good elements to choose from, i.e., a single robot cannot differentiate between the elements' qualities based on its local perception. This introduces the second scenario of the collective perception problem we investigate, called symmetry breaking. In this instance, there are multiple colours $n \geq 2$ with equal concentrations. Here the robot swarm has to break the symmetry, that is, it makes a consensus decision in favour of any present colour.

For the problem of symmetry breaking we study a symmetric best-of-n-problem, meaning all options have equal qualities. Similar to the adaptation problem, we assign a colour to each option and the colour's concentration represents the quality. Here we choose $n \in \{2, 3, 4, 5\}$ different colours with concentration $\rho_i = \frac{1}{n}$ respectively. In contrast to the problem of adaptation, we do not need to consider a dynamic environment for testing the swarm's ability of symmetry breaking. However we consider the swarm in an undecided state, which could be due to environmental change, e.g., colours with equal concentrations appear. Therefore, we initialise the swarm uniformly distributed over all present colours. With this situation we can study, if the robot swarm is able to break the symmetry towards any present colour.



(a) Top down view of the environment in simulation with 50 Kilobots.



(b) Top down view of the environment in reality with 1 Kilobot.

Figure 4.2: Adaptation environment implemented on the Kilogrid. There are $n = 2$ colours, with yellow having a higher concentration than blue ($\rho_b = 0.41 < 0.59 = \rho_y$). The size of the environment is 1 m × 2 m in reality and simulation.

Implementation of the Environment

The implementation of arbitrary best-of-n-problems need to allow the user to choose any number of options, i.e., colours, and their qualities, i.e., colour concentrations. For the implementation of the aforementioned symmetric and asymmetric instances of the best-of-n problem we use the augmented reality arena Kilogrid, described in Section 3.2. Thereby, we utilize the chessboard-like environment of the Kilogrid exploiting its structure comprised of cells and modules. We can assign any colour to any cell, thus we can have any number of options. The concentration of colour i and thus its quality is given by

$$\rho_i = \frac{\#\text{cells}_i}{684}, \tag{4.1}$$

with $\#\text{cells}_i$ being the number of cells which get colour i assigned. The concentration represents the relative quantity of colour i in the environment, thus we divide the $\#\text{cells}_i$ by the total number of coloured cells.
Following the concentrations, we randomly assign each cell (besides the wall cells) a colour, e.g., see Fig. 4.2. We use a uniform distribution for the random assignment.
In Section 4.3, we describe how the robots estimate the colour concentrations. This implementation fulfils all requirements to create arbitrary best-of-n-problems. The specific parameters used in the experiments are described in the experimental set-up sections of Chapter 5 and Chapter 6.

4.2 Robot Behaviour

For solving the above stated problem of collectively selecting the colour with the highest concentration out of n colours, we designed the individual robot behaviour. This robot behaviour is adapted from Talamali et al. (2021) to our specific environment. A detailed description of the algorithm can be found in Section 4.3. The main focus of design is to make the behaviour as minimalistic as possible, so that it can be run on minimal machines with limited capabilities.
Because the coloured cells are distributed over the entire environment, for monitoring and estimating the concentrations, it is sufficient that the robots do a random walk, with an integrated wall avoidance behaviour. During this random walk, the robot samples the ground periodically and integrates these samples over time to make an estimate of the concentration of the colour it wants to measure. After a fixed time interval of sampling, the robot updates its colour concentration estimate. This allows the robot to react to possible environmental changes, which is necessary assuming a dynamic environment. The concentration estimate update is either used to overwrite the current estimate, if the sampled colour is equivalent to the current commitment, or used for the colour's quality comparison, which will be explained below. Once the sampling of one colour is completed, the robot randomly selects a new colour to sample, in our case it selects the colour that it finds beneath itself. Sampling a random colour, allows the robot to explore new colours, which can be better than the colour the robot is currently committed to.
A robot updates its commitment either individually or socially. The individual update

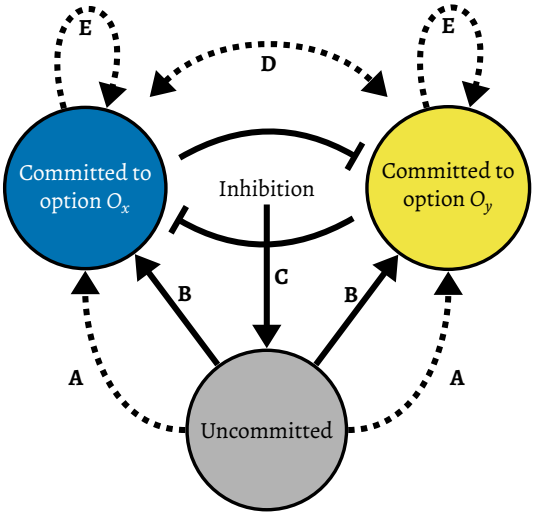


Figure 4.3: Commitment update state machine for $n = 2$ options. The dotted lines mark commitment updates based on self-sourced evidence and the solid lines mark commitment update based on social evidence. The uncommitted robot can commit through either discovery (A) or recruitment (B); committed robots change state through cross-inihibition (C) or direct switching upon comparison (D), and do not change their own state when re-estimating their own opinion (E).

is implemented as a comparison between the current ($\hat{\rho}_m$) and the newly estimated ($\hat{\rho}_e$) colour concentration, see Fig. 4.3 dotted arrows. If $\hat{\rho}_e > \hat{\rho}_m$ the robot adopts the new colour and changes its opinion.
For the social update, we extended the classical voter model by Holley and Liggett (1975). Similar to the classical voter model, the robot randomly selects one received message of its neighbours to update its commitment. The exchanged messages only contain the sender's (colour) commitment, bringing the advantages of limiting communication requirements among robots and preventing the spread of misinformation about option qualities, as the individual robot's estimates can be subject to high noise (Talamali et al., 2019; Parker and Zhang, 2004). If the received colour differs from the robot's own opinion, the robot starts to change its opinion based on the social interaction pattern, see solid arrows Fig. 4.3. We chose the cross-inhibition social interaction pattern (Pais et al., 2013; Reina et al., 2015; Talamali et al., 2019). In this pattern the robot switches into an uncommitted state, if it receives a colour different to its opinion. When switching into the uncommitted state, the robot forgets its previous commitment, thus stops disseminating it and starts sampling a random colour (the colour beneath itself). After finishing the sampling process the obtained information ($\hat{\rho}_e$) can be used for a self-sourced commitment switch. In this special case, $\hat{\rho}_m = 0$, thus the robot would always switch to the self-sourced information. Robots in the uncommitted state continue to listen to incoming messages, which allows the robot to be recruited. Recruited robots adopt the received colour and start sampling it for making an estimate of its concentration.
Once the robot is committed and has an estimate of the colour's concentration it is com-

mitted to, the robot broadcasts its opinion with a frequency linearly proportional to the estimated quality (Talamali et al., 2021; Parker and Zhang, 2009). The quality dependent communication was inspired by house-hunting behaviour of social insects (Franks et al., 2002; Jeanson, Dussutour, and Fourcassié, 2012) and was successfully implemented in several swarm robotics systems (Reina et al., 2015; Valentini et al., 2016a; Parker and Zhang, 2004; Parker and Zhang, 2009; Valentini, Hamann, and Dorigo, 2014).
Besides the changes in how the robot samples, due to the new environment, we included a mechanism to gradually increase speaking, further explained in Section 4.3. We included this mechanism, because we faced the problem of many robots being inactive, either uncommitted or recently recruited but without a complete sampling of the alternative. During this period a very small minority of robots, especially in case of global communication, can vote for their colour, which can create the dangerous situation in which a single robot can recruit the whole swarm and thus reduce the benefits of the collective intelligence.

4.3 Kilobot Controller

This section presents a detailed description of the implementation of the robot behaviour from Section 4.2. Thereby, we emphasise the adjustments we had to do for the new environment and the new mechanism of gradually increase speaking.
We begin with an overview of the main loop of the Kilobot controller. Afterwards, we explain the routines of motion, sampling, updating commitment and communication which are executed in parallel.

Main Loop

As mentioned in Section 4.2, the controller for the Kilobot extends the controller presented in Talamali et al. (2021).
The main loop of the robot controller is executed approximately every 32 ms. Tab. 4.4 lists the most important parameters of the controller.
The robot stores two pieces of information, the colour the robot is committed to, i.e., an option o_i, and the estimated colour's concentration, i.e., the estimate of the option's quality $\hat{\rho}_m$. To save memory, we represent a colour as an integer, which represents the robot's opinion. The value 0, thereby means the robot is uncommitted and the colours are consecutively numbered, e.g., $o_1 = 1$ for yellow, $o_2 = 2$ for blue etc. Further, we introduce an uninitialised opinion $i = 20$, which is the default value when starting the robot controller. This value indicates to the robot that it needs to wait for the *initial message*, which overwrites the current commitment to the for the experiment appropriate value. Also, it is used to check if all robots are initialised correctly, i.e., each robot received the *initial message*.
The estimated concentration is encoded as a value in $[0, 1] \in \mathbb{R}$. Thereby, 0 is the lowest possible and 1 is the highest possible, for the robot measurable, concentration.

Before the main loop is executed the `setup` function is called, similar to the Kilogrid

Table 4.4: Overview of the most important controller parameters. If it is a fixed parameter, which does not change during any experiments the value of it is assigned next to the unit.

Name	Explanation	Unit
o_i	Colour the robot is committed to.	$\mathbb{N}$
$\hat{\rho}_m$	Estimated concentration of the colour the robot is committed to.	$\mathbb{R}$
uID	Unique identifier.	$\mathbb{N}$
τ_s	Sampling time, $\tau_s = s\delta_s$.	seconds
τ_u	Update time.	2 seconds
τ_b	Broadcast time.	0.5 seconds
s	Number of samples.	$\mathbb{N}$
δ_s	Time between samples s.	seconds
r_c	Communication range.	cells

module. We use this `setup` function to initialise the random seed and the counters of the Kilobot controller, which we will mention in more detail at the appropriate parts of this sections. After initialising the robot, it waits for the *initial message*, which contains experiment dependent parameters, see Tab. 3.8. After receiving this message and setting the experiment dependent parameters, the robot selects its first way point for its random walk, explained later in more detail. Finally the *initial message* allows the robot to start the main loop.

In Fig. 4.5, we present the procedure of the main loop. As in the Kilogrid modules, the message reception is implemented as a callback function. Hence, new messages can arrive at any point in time during the loop execution. To ensure that the calculations of the Kilobot are consistent, the received message is stored temporary to not overwrite currently used values. To access the latest data, the first step of the main loop is to update the received message data, by overwriting the variables used for calculation. The robot sequentially checks for a new grid message followed by checking for a new agent message.
During the grid message update, the robot refreshes its information about the role of the underlying cell, its x and y coordinates and the cell's option. Further, if the robot moved to another cell it calculates its current orientation by

$$\phi = \frac{180}{\pi} \operatorname{atan2}\left(\Delta_y, \Delta_x\right), \tag{4.2}$$

with ϕ the robot's orientation in degrees and Δ_x and Δ_y the current robot position minus its previous position in cells. With the estimation we can calculate the robot's orientation

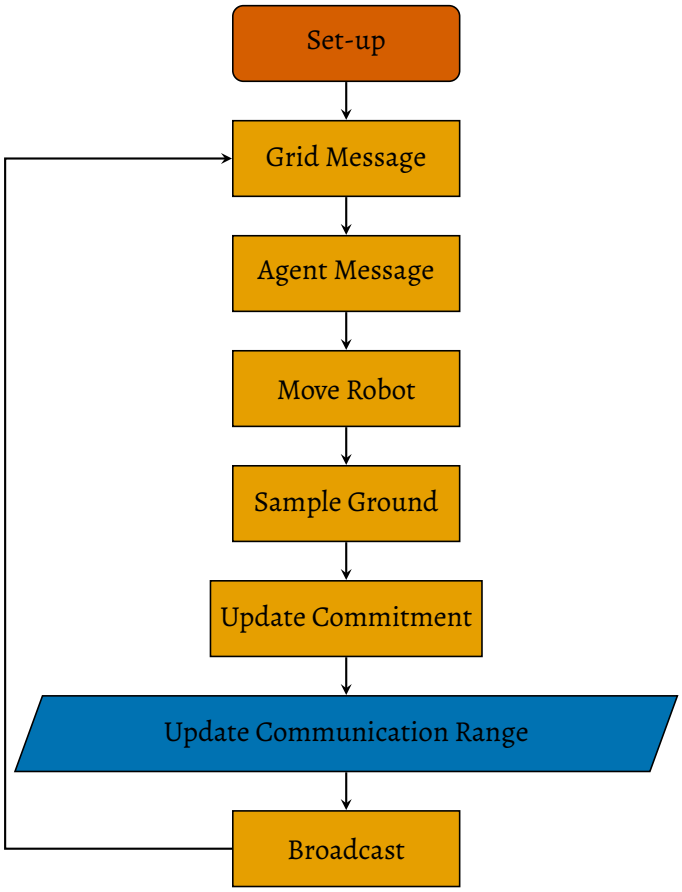


Figure 4.5: This Flow chart shows the main loop of the robot controller. The orange steps are always executed. The extension of updating the communication range, in blue, is discussed in Chapter 6.

with a precision of 45°, which is sufficient for the Kilobot's random walk and wall avoidance.
Next, the robot checks for new received messages from neighbours. While checking for a new message, we ensure that the robot does not receive its own message, by comparing with the sending robot's unique identifier. When receiving a message from a neighbour a notification flag is set for the update commitment routine, marking the presence of a new social information. If the robot received multiple messages from different neighbours during one main loop cycle, the last message is chosen for the social information, as the robot only has a buffer size for storing one infrared message.

Motion

The motion routine is independent form the other routines, that is, the movement of the robot is not influenced by its opinion, social clues nor sensory information (reading the floor colour). The motion routine consists in a random walk in the environment and a wall avoidance behaviour. For practical reasons, because the Kilobot's motion is not reliable (prone to drive circles and does not properly explore the environment), see Section 3.1, we

implemented the random walk and the wall avoidance with additional information, e.g., access to positional information in a global frame. However, the current random walk and wall avoidance can be replaced by any other (less demanding) algorithm of random diffusion.
For the random walk, we implement a random waypoint mobility model (Bettstetter, Hartenstein, and Pérez-Costa, 2004). Through the random waypoint mobility model, the robot selects a random position in the environment, which it sets as its target destination. Once the destination is reached, the robot selects a new random target destination, which needs to be at least 20 cm away from its predecessor. We also implemented a time out interval of 2 min, after which a new random target destination is selected to avoid robots getting stuck at traffic jams caused by groups of robots moving in opposite directions, or robots not moving due to malfunctioning motors. The robot avoids collisions with the walls surrounding the environment by selecting random destinations that are at least three robot-body lengths (approximately 10 cm) far from the walls. As the movement can be subject to high levels of noise, it still can happen that the robots arrive near the walls. Once it gets at a distance smaller than three robot-body lengths from any wall, the robot starts the wall avoidance manoeuvre by rotating away from the wall towards the center of the environment, followed by moving straight for a few seconds. Due to missing proximity sensors, we cannot implement any obstacle avoidance manoeuvre to prevent colliding robots.

Sampling

The next routine executed in parallel is the sampling routine. It consists in acquiring information from the environment about the concentration of one option, i.e., a colour. One cycle of the sampling routine takes $\tau_s = s\delta_s$ seconds, where s is the number of samples to take required for making an estimation on the colour's concentration and δ_s is the time between taking two samples. The process of obtaining one sample (sampling step) is visualised in Fig. 4.6.
Each sample is a binary value that indicates the presence or absence of the option o_i of interest, in our case, the robot checks if the cell beneath itself is of a given colour. The robot tries to do a sampling step every δ_s seconds. If the robot is on a wall tile, it skips the sample and the sampling counter does not get incremented.
Next, the robot checks if it reached the required number of samples s for making an estimate $\hat{\rho}_e$. In case of not having enough samples collected the robot samples the current ground as described above. When the robot reached the required number of samples, it estimates the quality of the sampled option as

$$\hat{\rho}_e = \frac{\#\text{samples of } o_i}{s}.$$

If the colour the robot was sampling is equivalent to its current commitment, it updates its concentration estimate, arrow E in Fig. 4.3. In case the sampled colour differs from the current commitment, the concentration estimate is used in the commitment update routine, as new self-sourced information, which can be used for a commitment update, arrow D in Fig. 4.3.

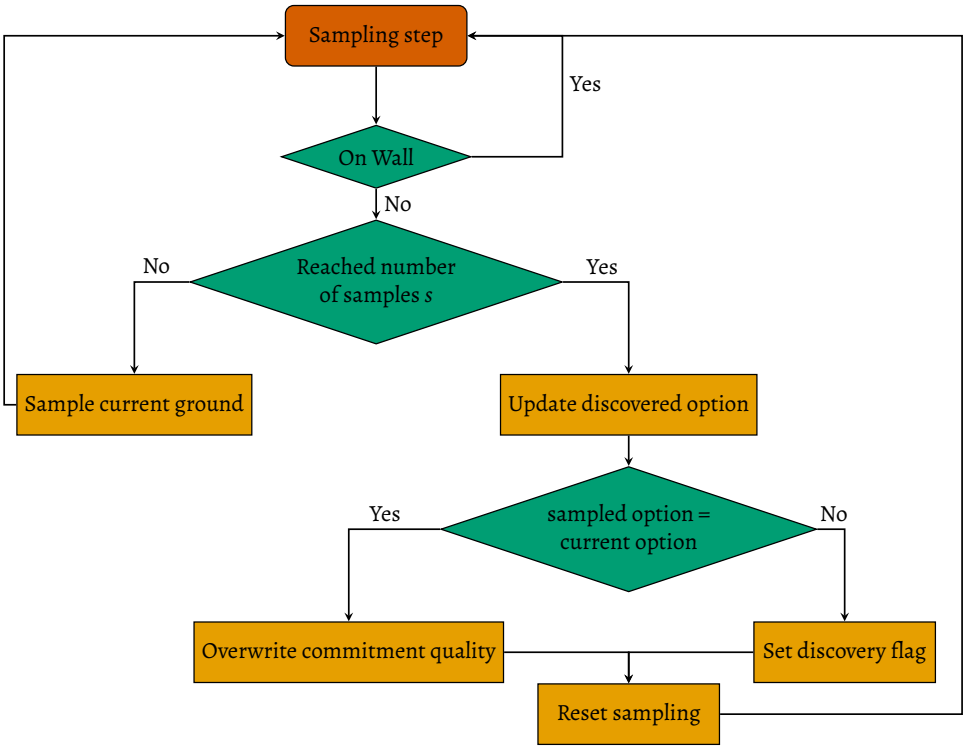


Figure 4.6: This Flow chart shows the process of taking one sample. This process is done every δ_s seconds and repeated s times before the robot can make an estimate on the concentration it currently samples.

A new sampling cycle starts when the previous sampling cycle collected s samples, or when the robot changes opinion through social evidence. In the former case and when the robot becomes uncommitted, the robot determines which colour to sample randomly, in our case it selects the colour of the ground beneath itself; instead when the robot becomes committed, it starts sampling the colour which it has been recruited to.

Update Commitment

The robot commitment update routine is executed approximately every $\tau_u = 2$ s. To avoid cascades caused by synchronous robot updates we add uniform distributed noise to τ_u ($\pm$50 ms or $\pm$5%).
As mentioned in Section 4.2, the robot can update its commitment either through social or self-sourced information. In the previous section we show how the robot gathers self-sourced information. This information is taken into account, if the newly estimated quality $\hat{\rho}_e$ is larger than the current quality estimate $\hat{\rho}_m$. Social information, collected from neighbouring robots, is considered for the updating process if the received colour differs from the robots current commitment. If both sources are available the robot chooses one at random and discards the other, else the robot uses the available source of information. In the case of commitment update based on self-sourced information, arrows A and D in Fig. 4.3, the robot adopts the new option and the quality estimation it made earlier

($\hat{\rho}_m \leftarrow \hat{\rho}_e$). Thus, the robot can directly speak at full frequency.
Updating the commitment based on social information implements the social interaction pattern cross-inhibition presented in Section 4.2. The next steps depend on the robot's commitment. For the case the robot is uncommitted, it gets recruited to the received option, and starts sampling it, arrow B in Fig. 4.3. During this sampling process the robot uses gradually increasing speaking for increasing the swarms stability. In the other case, the robot is committed to any option, it gets cross-inhibited arrow C in Fig. 4.3, which means the robot becomes uncommitted.

Broadcasting

At the end of the control loop, the broadcasting routine is executed. In this routine the

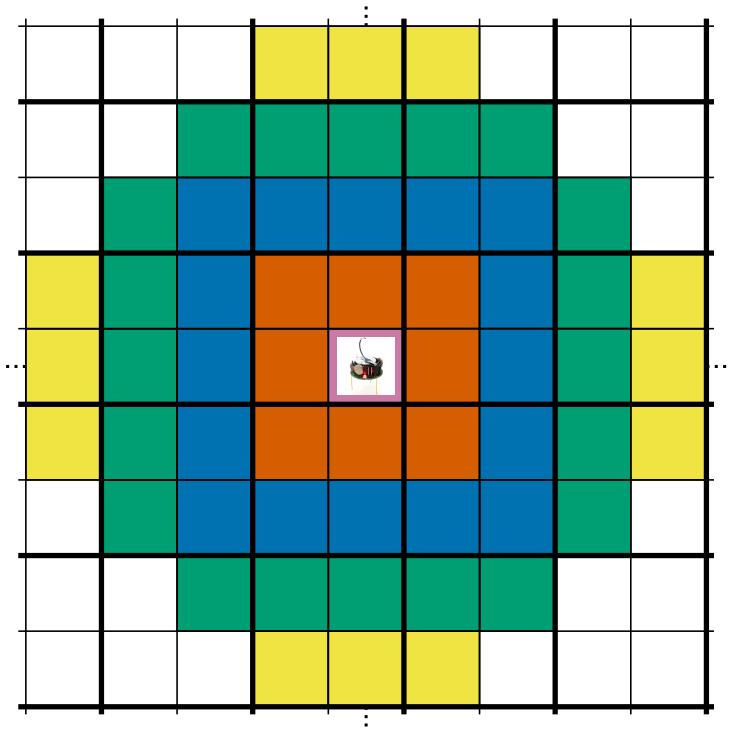


Figure 4.7: Different communication radii of the Kilobot utilising the Kilogrid for communication. Thereby, purple $r_c = 1$, red $r_c = 2$, blue $r_c = 3$, green $r_c = 4$ and yellow $r_c = 5$. The euclidean distance measure is used.

robot tries to broadcast its commitment up to every $\tau_b = 0.5$ s. This value is a hardware limitation, as the robot cannot reliable transmit messages in a higher frequency, as described in Section 3.1.
As mentioned in Section 4.2, the communication rate is proportional to the option's quality estimate of the robot $\hat{\rho}_m$. It is implemented such that the robot tries to send a message every 0.5 s, with a probability of

$$p = \begin{cases} 2\hat{\rho}_m & \text{, if } \hat{\rho}_m < 0.5 \\ 1 & \text{, else} \end{cases} , \tag{4.3}$$

where $\hat{\rho}_m$ is the robots estimated quality. We introduce the scaling factor 2, because when the colour's concentration is higher than 50% it presents the absolute majority and should be treated as the best alternative. For lower concentrations the probability p scales linearly between 0 and 1, thus the robot controller performs better in scenarios with multiple options as the concentrations are lower due to normalisation.
Note that, while in the case of $n = 2$ a concentration lower than 50% implicitly indicates the predominance of the other colour, this reasoning does not generalise to $n > 2$ and therefore we do not consider this deductive mechanism. When the robot has just been recruited and does not have an concentration estimate yet, it gradually increments its communication frequency, described in more detail in the next section.
The broadcast message is then distributed with a communication range r_c, see Fig. 4.7. We apply the same distance measure as in Eq. (3.1) and the transmission is done via the Kilogrid as described in Section 3.2. This allows the robot to choose its communication range in approximately 5 cm steps. The communication range in cm is given by

$$r_{\text{cm}}(c) \approx 2.5 + 5(c - 1), \tag{4.4}$$

where c is the number of cells the robot wants to forward its message.

Increasing Speaking

The mechanism of gradually increasing speaking allows committed robots, which have not finished sampling and thus not have an estimate of their commitment yet, to participate in the voting process.
We included this mechanism because we noticed that during the experiments several robots where recruited to either colour, but have not finished sampling. This opens up the possibility for a small minority of robots, in the most extreme case one robot with global communication range, to influence all the other robots towards its commitment. This makes the swarm prone to errors and in general more unstable.
We illustrate the process in Fig. 4.8. Gradually increasing speaking applies during the

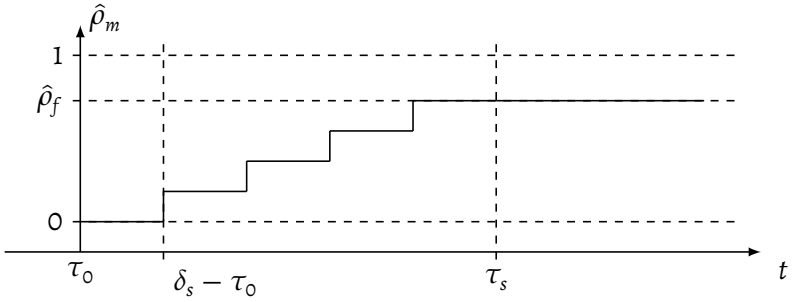


Figure 4.8: When the robot does not have a concentration estimate $\hat{\rho}_m$ of its colour, it uses a provisional estimation. τ_0 is the moment the robot got recruited to either option. Here the robot samples $s = 6$ times every δ_s seconds. Besides the first sample the robot always samples the colour it is currently sampling, arriving at a final estimate of $\hat{\rho}_f$.

process of sampling after recruitment, arrow B in Fig. 4.3. We utilise that the robot al-

ready obtained some information about the environment, i.e., the samples of the currently running sampling cycle. This information can be used to do a provisionally estimate of

$$\hat{\rho}_m = \frac{\#\text{currently collected samples of the option to sample}}{\#s},$$

which otherwise would be 0. Thus the robot is able to speak during the sampling process, see Eq. (4.3).
This mechanism makes a conservative estimation of the current colour concentration estimation, meaning it never overestimates the final concentration estimation and is very low at start and gradually increases; through this mechanism recruitment takes time.

5

Experiments

This chapter covers the conducted experiments for the adaptive best-of-n scenario and the symmetry breaking scenario. We start by describing the metrics used for evaluation, followed by describing the experimental set-up. Finally, we present and discuss the results.

5.1 Metrics

In all experiments we track the commitment state of each Kilobot of the robot swarm at each simulation step. These tracking data are subject to high fluctuations, as you can see in Fig. 5.1[5] lighter lines. Therefore, we apply a moving average filter on the collected data to improve data visualisation. We choose a window size of 30 s, see Fig. 5.1 thicker lines. At the border the window is truncated, means the average is taken over only the elements that fill the window. This ensures that we can apply the following metrics without risking that noise impairs our results.

Adaptation

In Section 4.1, we describe the adaptive best-of-n scenario, which tests the robot swarm's ability of adapting to environmental changes. Thereby, we assume $n = 2$ options, with $\rho_1 < \rho_2$. We say the swarm adapted successfully if the large majority of robots (70% of the robot swarm) is committed to the superior option (w.l.o.g. colour yellow) for at least 5 minutes. This threshold helps to avoid counting short-lived random fluctuations as successful adaptations as we want a stable majority. The adaptation probability calculates as the number of successful runs (the swarm adapted) divided by all runs for the given parameter set.
The adaptation time is measured as the moment when the subpopulation, committed to the superior option, reaches 70% at the beginning of the 5 minute time interval. For the adaptation time we only consider successful runs.

[5] We generated the data plots using the matlab2tikz, Nico Schlömer (2022). matlab2tikz/matlab2tikz (https://github.com/matlab2tikz/matlab2tikz), GitHub. Retrieved February 15, 2022.

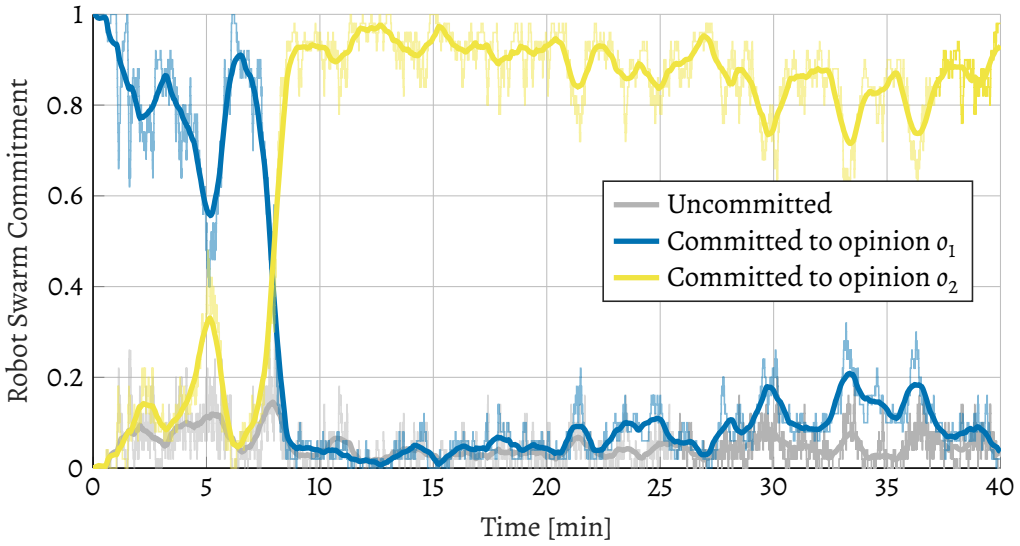


Figure 5.1: Example of collected data of one adaptation experiment. The raw data are depicted as the thin lines. For the thick lines we applied a moving average filter to denoise the data. We chose this specific run to show our intention: through minute 30 to 40 the raw data momentarily drop below 70% even though the consensus is stable.

Symmetry Breaking

For testing the robot swarm's ability to break the symmetry, we use the symmetry breaking scenario presented in Section 4.1. In the symmetry breaking scenario, we assume multiple options $n \in \{2, 3, 4, 5\}$, with equal quality. We say a swarm successfully breaks symmetry if the large majority of robots (70% of the robot swarm) are committed to any of the present options. Similar to the adaptation metric, we require the majority to hold for at least 5 minutes to avoid counting short-lived fluctuations as symmetry broken. The symmetry breaking probability calculates as the number of successful runs (the swarm broke the symmetry) divided by all runs for the given parameter set.
The symmetry breaking time is measured as the time the fraction of the robot swarm committed to any option reaches 70% in the beginning of the 5 minute time interval. For the symmetry breaking time we only consider successful runs.

Problem Difficulty κ

We model the problem difficulty for the adaptive best-of-n scenario by varying the colour concentrations, as mentioned in Section 4.1. Therefore, we use the definition as in Talamali et al. (2019),

$$\kappa = \frac{\rho_l}{\rho_h}, \tag{5.1}$$

with ρ_l the lowest concentration and ρ_h the highest concentration present in a given environment. This leaves us with an interval $\kappa \in [0, 1]$, where $\kappa = 1$ presents the case of symmetry breaking, because all colours have equal concentration and for smaller κ the

problem becomes easier, e.g., for $\kappa = 0$ there is only one colour in the environment. Options with quality in between are ignored. An overview of the colour concentration for different κ is given in Tab. 5.2.

Table 5.2: Overview of the cell concentration for different κ, assuming $n = 2$ colours. The total number of cells is 684.

κ	$\#\text{cells}_l$ / ρ_l	$\#\text{cells}_h$ / ρ_h
0.7	282 / 0.41	402 / 0.59
0.8	304 / 0.44	380 / 0.56
0.9	324 / 0.47	360 / 0.53
0.95	333 / 0.49	351 / 0.51

Switches

We define a switch as the majority of the swarm changes opinion, that is, when more than 50% of the swarm is committed to a different colour than the simulation step before. Thereby, during the adaptation experiments, we only count switches towards the inferior option (w.l.o.g., option o_1 or color blue). Counting only switches towards option o_1 provides a clearer analysis, as the initial switch from option o_1 to option o_2 is desired, because it is necessary for the process of adaptation, while switches in general are not desired as they indicate highly unstable system dynamics. Further, by counting only half of the switches we do not lose information because in environments with $n = 2$ options, we can compute from the number of switches from o_2 to option o_1, the number of switches in the other direction, from option o_1 to option o_2.
For the case of symmetry breaking, we generalise the aforementioned concept to not counting the first switch, i.e., when the swarm commits the first time to any option with more than 50% of the swarm members, as it is necessary for the process of symmetry breaking, but we count all the subsequent switches.

5.2 Adaptation

In this section, we test the ability of adaptation of the robot swarm. We begin so by describing the experimental set-up, followed by the results. At the end of this section, we discuss the results.

Experimental Set-Up

We propose an asymmetric best-of-n problem with $n = 2$ options, as described in Section 4.1, for testing the robot swarm's ability to adapt to environmental changes. We test the robot behaviour for different problem difficulties $\kappa = \{0.7, 0.8, 0.9, 0.95\}$. This set of problem difficulties was chosen because for $\kappa < 0.7$ we found that the robot swarm is

able to adapt for all tested parameter sets and $\kappa = 1$ is studied in the other scenario.
We model the dynamics of the environment as an instantaneous change of colour concentration for testing the ability of adaptation, further described in Section 4.1. There, we also described the initial condition of the robot swarm, that is w.l.o.g. fully committed to option o_1 (blue) with a high quality estimate of $\hat{\rho}_m = 0.8$. For a realistic state of the swarm, we initialise all timers and the colour the robots currently sample randomly. Each simulation run simulates 40 minutes (76800 simulation steps, with 32 ms per simulation step).
Our study focuses on two key parameters, which are essential for the robot swarm's dynamic, that are the sampling time and the connectivity of the robot swarm. Both parameters can be precisely chosen thanks to the environment and the experimental set-up, which was not possible before.
The first key parameter, the sampling time τ_s, varies the time the robots need to form a concentration estimate. Hence, τ_s varies the delay for recruited robots to recruit robots themselves, because the robots first need to make a concentration estimate before they can recruit other robots towards their opinion. This delay is a necessary condition for triggering the *less is more* effect, as described in Section 2.1. The parameter τ_s is defined as $\tau_s = s\delta_s$ seconds. Therefore, we conduct sampling time experiments, where we vary the number of samples $s = \{5, 10, 15, 20, 25, 30, 45, 60\}$ while fixing $\delta_s = 1$ s and varying $\delta_s = \{0.5, 1, 2, 3, 4, 5, 6, 7\}$ s while fixing $s = \{5, 15\}$.
The second key parameter is the connectivity of the robot swarm, i.e., the average number of neighbours that every robot has. This parameter is influenced by the communication range of the individual robot and by the swarm density. Our experimental set-up allows us to vary the communication range $r_c = [1, 45]$ cells, while having a fixed swarm size of $N = 50$ robots. The swarm density is defined as robots per area, thus by increasing the swarm size N, while keeping the area fixed, we increase the swarm density. We vary the swarm size $N = [50, 500]$, while we fix the communication range to $r_c = 3$ cells.
Finally, for testing the influence of noise in our system, that is due to the spatial correlation of the cells, we also experiment with different noise levels. This is done by letting the robots sample their sensor readings from a normal distribution $\mathcal{N}(\mu, \sigma)$ with the mean μ based on the ground truth of the environment, $\mu = \rho_i$ for option o_i. We test different noise levels $\sigma \in \{0, 0.1, 0.2\}$.
We conducted four adaptation experiments, which we present in the next section.

Results

The first experiment studies the relationship between the two key parameters of the system, the sampling time and the connectivity of the robot swarm. We vary the sampling time by varying the number of samples s and keeping fixed the time between samples $\delta_s = 1$ s. We vary the communication range r_c with a fixed swarm size $N = 50$. The results for different problem difficulties κ can be seen in Fig. 5.3. The general trend is that the swarm is able to adapt, if either the communication range r_c is small and the sampling time is long or the communication range is large and the sampling time is short. If the communication range is chosen between $10 \leq r_c \leq 25$ and long sampling time, the swarm is really unlikely to adapt. However, the adaptation probability goes up again for

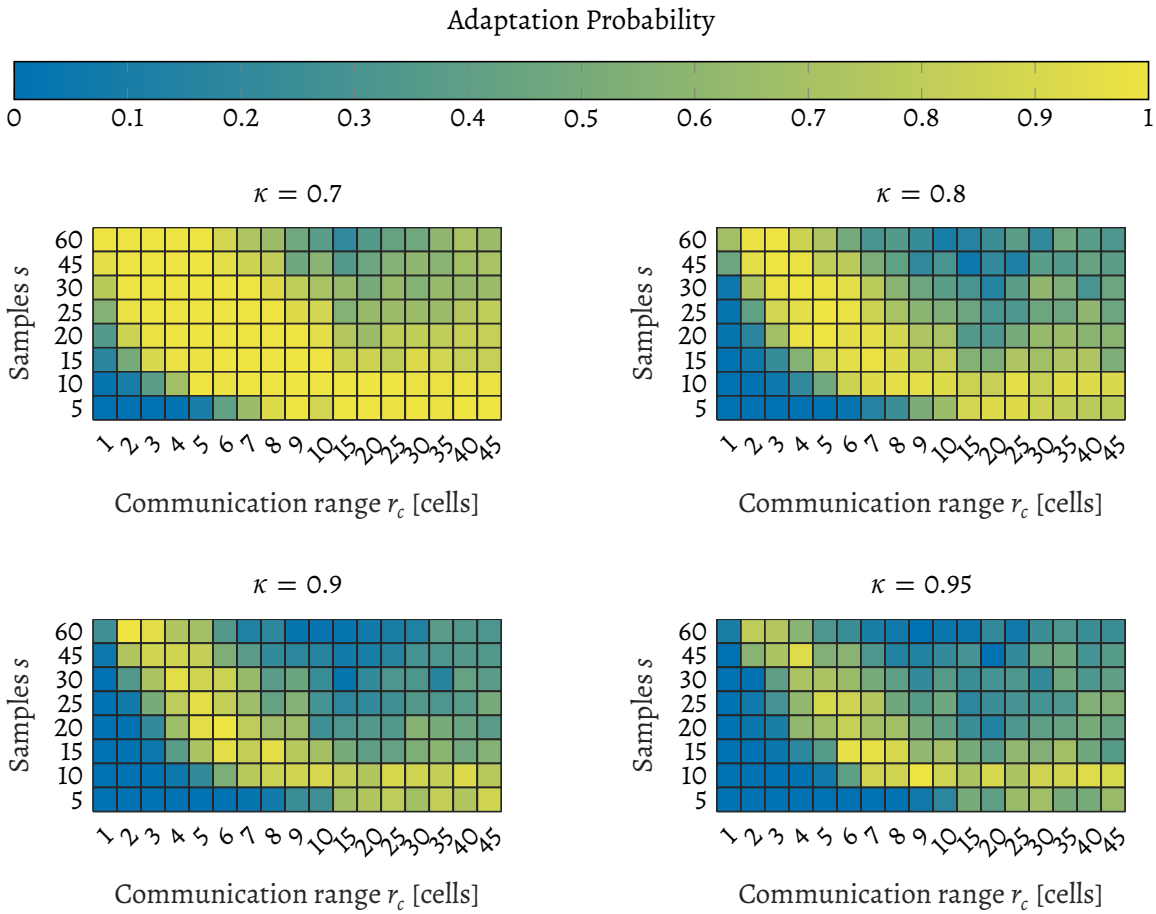


Figure 5.3: First adaptation experiment with varied static communication ranges $r_c = [1, 45]$ cells over different number of samples $s = [5, 60]$, with fixed $\delta_s = 1$ s for different problem difficulties $\kappa = [0.7, 0.8, 0.9, 0.95]$. The swarm size is $N = 50$. We conducted 40 independent simulation runs per data point.

larger communication ranges. This trend is visible in all problem difficulties. For simpler problems, e.g., $\kappa = 0.7$ in Fig. 5.3, the robot swarm is able to adapt with a wider range of parameters than for more difficult problems. The average switches of this experiment can be seen in Fig. 5.4. There are different scales, depending on the problem difficulty. The general trend is, that if the communication range is small and the sampling time is short, the robot swarm switches more often. For a more difficult problem, the swarm switches more often. For each data point we conducted 40 independent simulation runs. For further adaptation experiments, we decided to fix the problem difficulty to $\kappa = 0.9$, as it shows the trends the clearest.

In the second experiment, we again study the relationship between the two key parameters of the system. Different to the first experiment is, that now we vary the sampling time τ_s by increasing the time between samples δ_s and keeping the number of samples

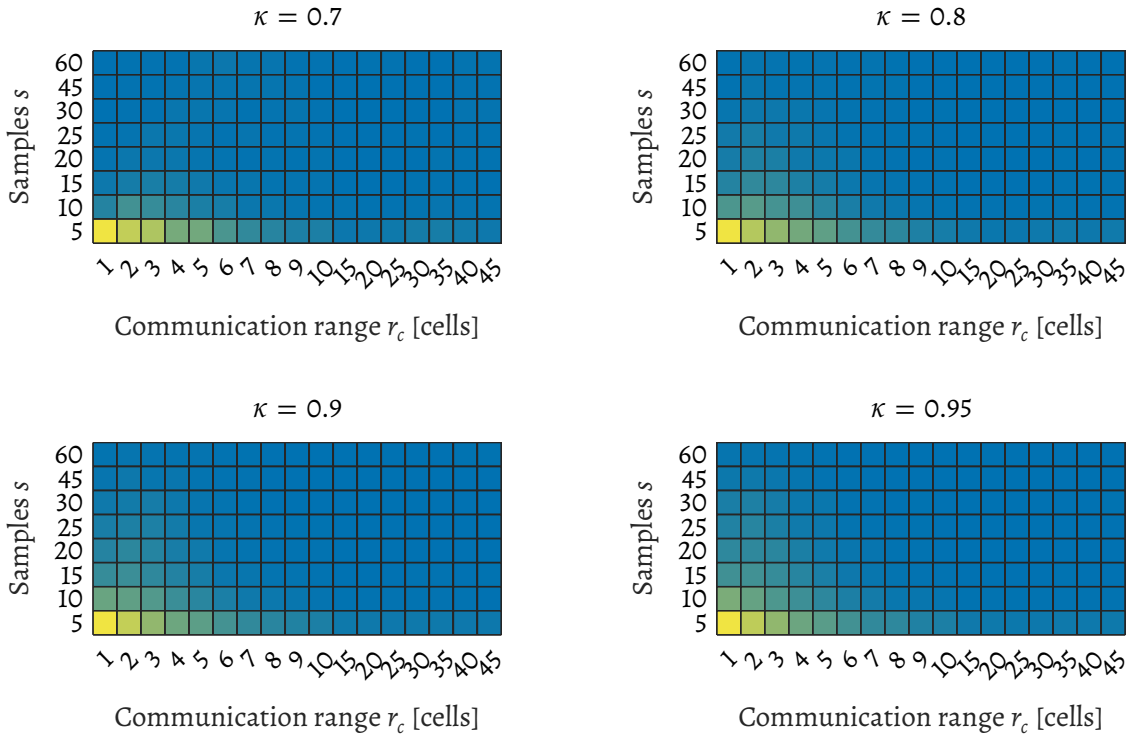


Figure 5.4: Average switches of the first adaptation experiment with varied static communication ranges $r_c = [1, 45]$ cells over varied number of samples s. The switches follow different scales: $\kappa = 0.7$ the maximum number of switches (yellow) are 11, $\kappa = 0.8$ the maximum number of switches are 30, $\kappa = 0.9$ the maximum number of switches are 40 and $\kappa = 0.95$ the maximum number of switches are 46. We conducted 40 independent simulation runs per data point.

$s = 15$ fixed. The results for $\kappa = 0.9$ can be seen in Fig. 5.5. The adaptation dynamics are similar to the first experiment, that is, the swam is able to adapt with small communication range and a long sampling time, now enforced by longer time between samples, or large communication ranges and shorter sampling time. We chose the swarm size $N = 50$ robots. For each data point we conducted 30 independent simulation runs.

In the third experiment, we test our system's ability of adaptation for different noise levels. We fix the number of samples $s = 5$ and vary the communication range r_c over time between samples δ_s. Further, we apply different levels of noise $\sigma \in \{0, 0.1, 0.2\}$. The results are shown in Fig. 5.6. In the absence of noise the swarm is able to adapt for all cases, besides large communication ranges and long sampling time. If there is noise, the swarm loses the ability of adapting for small communication ranges and short sampling time. For each data point we conducted 30 independent simulation runs with $N = 50$ robots.

For the fourth and last adaptation experiment, we varied the swarm size N over different number of samples s. We fixed the communication range to $r_c = 3$ cells ≈ 12.5 cm and

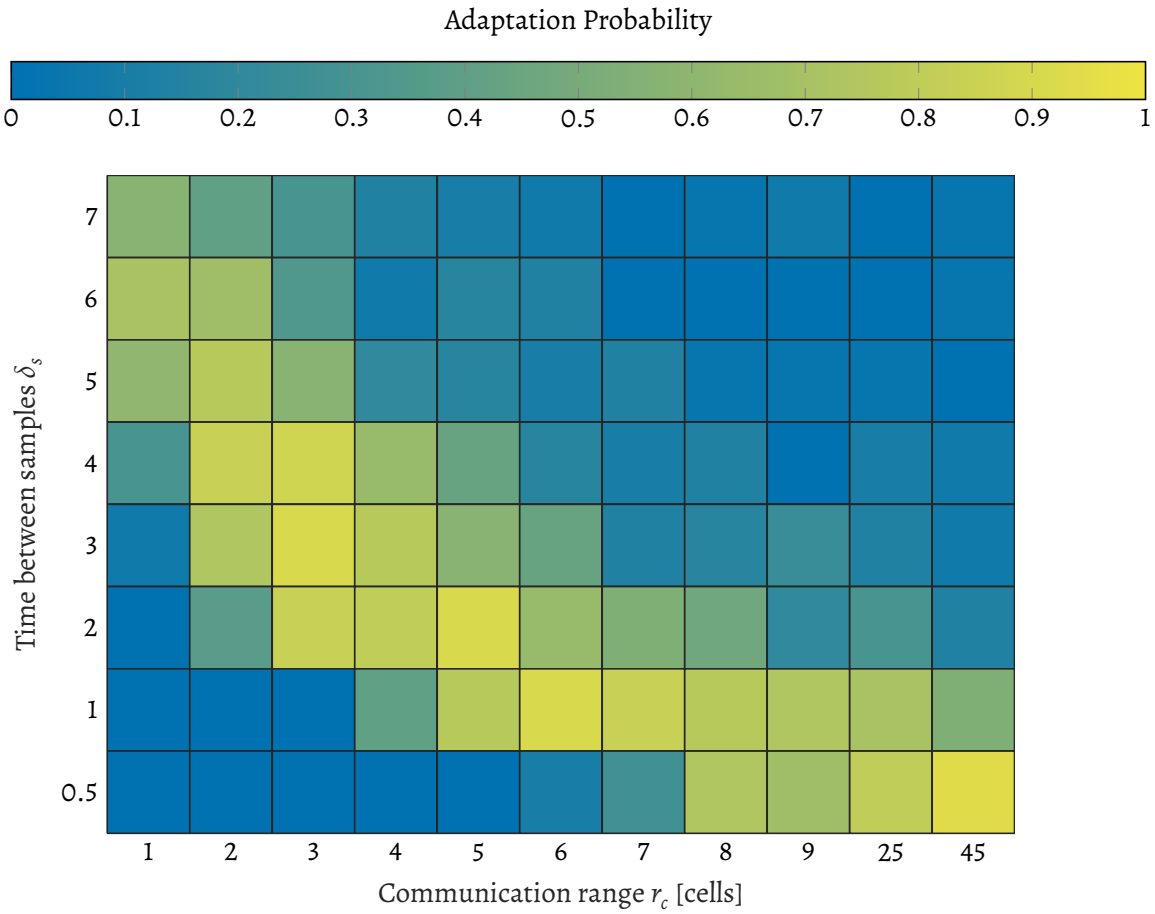


Figure 5.5: Second adaptation experiment varying static communication ranges $r_c = [1, 45]$ cells over time between samples δ_s for a fixed number of samples $s = 15$. The problem difficulty is $\kappa = 0.9$. We conducted 30 independent simulation runs with $N = 50$ robots per data point.

the time between samples to $\delta_s = 1$ s. The results for different problem difficulties can be seen in Fig. 5.7. The fourth experiment shows similar dynamics to the first and second experiment. The robot swarm is able to adapt for low density and longer sampling time or higher density and shorter sampling time. For simpler problem difficulties, the swarm is able to adapt with a larger range of parameters. The computations became very expensive, because the implementation of the Kilobot plug-in in ARGoS 3 creates one process for each Kilobot controller. This leads to problems with some script operating the cluster nodes, because to many processes are spawned, resulting in a high probability of crashing the cluster nodes, that is why we only did 10 independent simulation runs per data point[6].

[6] Special thanks to Jonas Kuckling (cluster administrator at IRIDIA), for still allowing me to run experiments on the cluster.

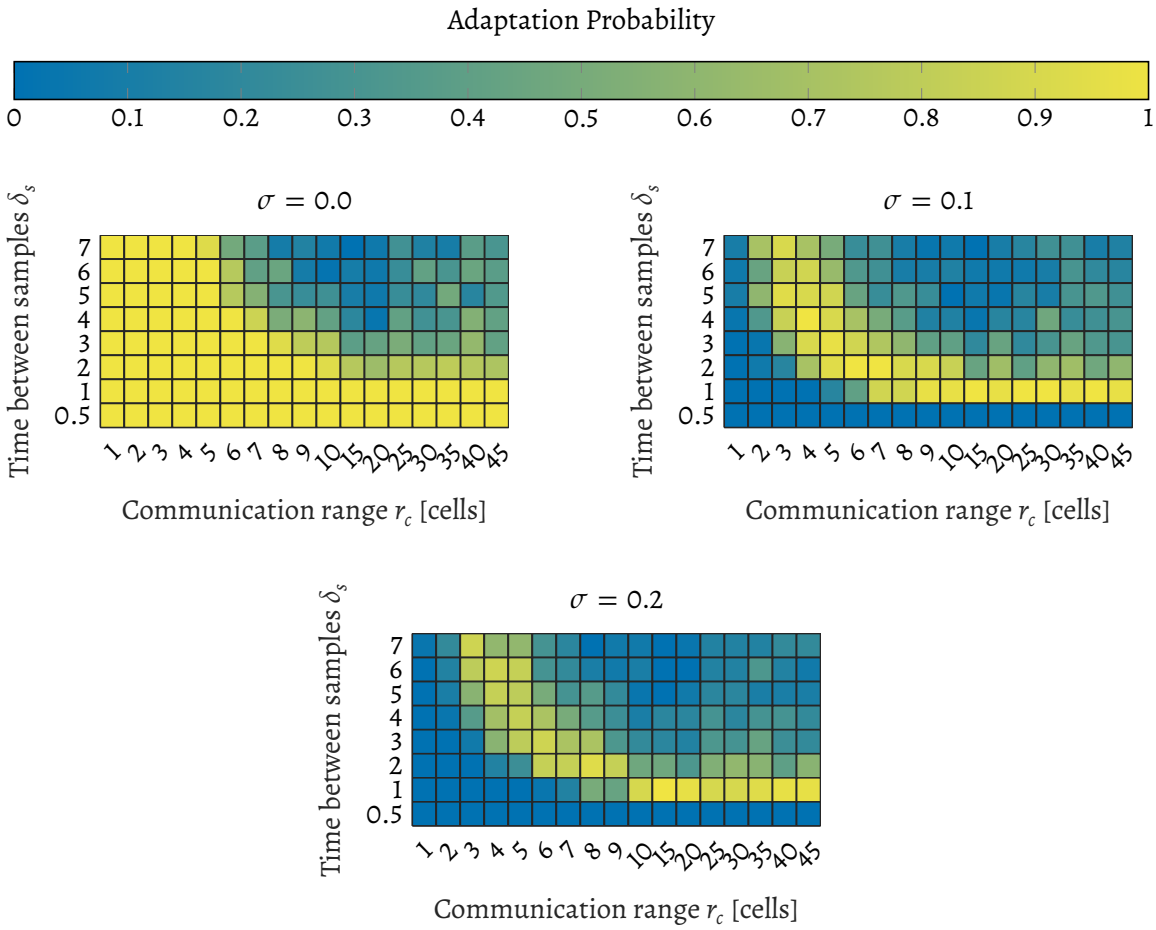


Figure 5.6: Third adaptation experiments with different noise levels. We vary the communication range $r_c = [1, 45]$ cells over different time between samples $\delta_s = [0.5, 7]$ s, while fixing the number of samples to $s = 5$. The problem difficulty is $\kappa = 0.9$. We conducted 30 independent simulation runs with $N = 50$ robots per data point.

Discussion

In the first adaptation experiment, we varied recruitment time, through the number of samples s, and the connectivity of the robot swarm, through different static communication ranges r_c. Varying both key parameters is only possible due to the new implementation of the problem. The new way of sampling allows us to precisely choose the sampling time and thus the recruitment time. In the previous work of Talamali et al. (2021) the sampling time (recruitment time) depends on the relative position of the robot to the site or robot congestions, which may block the way, thus it is impossible to study different recruitment times. Further, the Kilogrid allows for choosing different communication ranges, which allow us to study the connectivity of the robot swarm. The first observation is that, if the problem becomes easier, the robot swarm adapts for a wider range of parameters. That is the reason why we chose the problem difficulty in the range of $\kappa = [0.7, 0.95]$. For simpler problems, e.g., choosing $\kappa = 0.6$ the robot swarm is able to

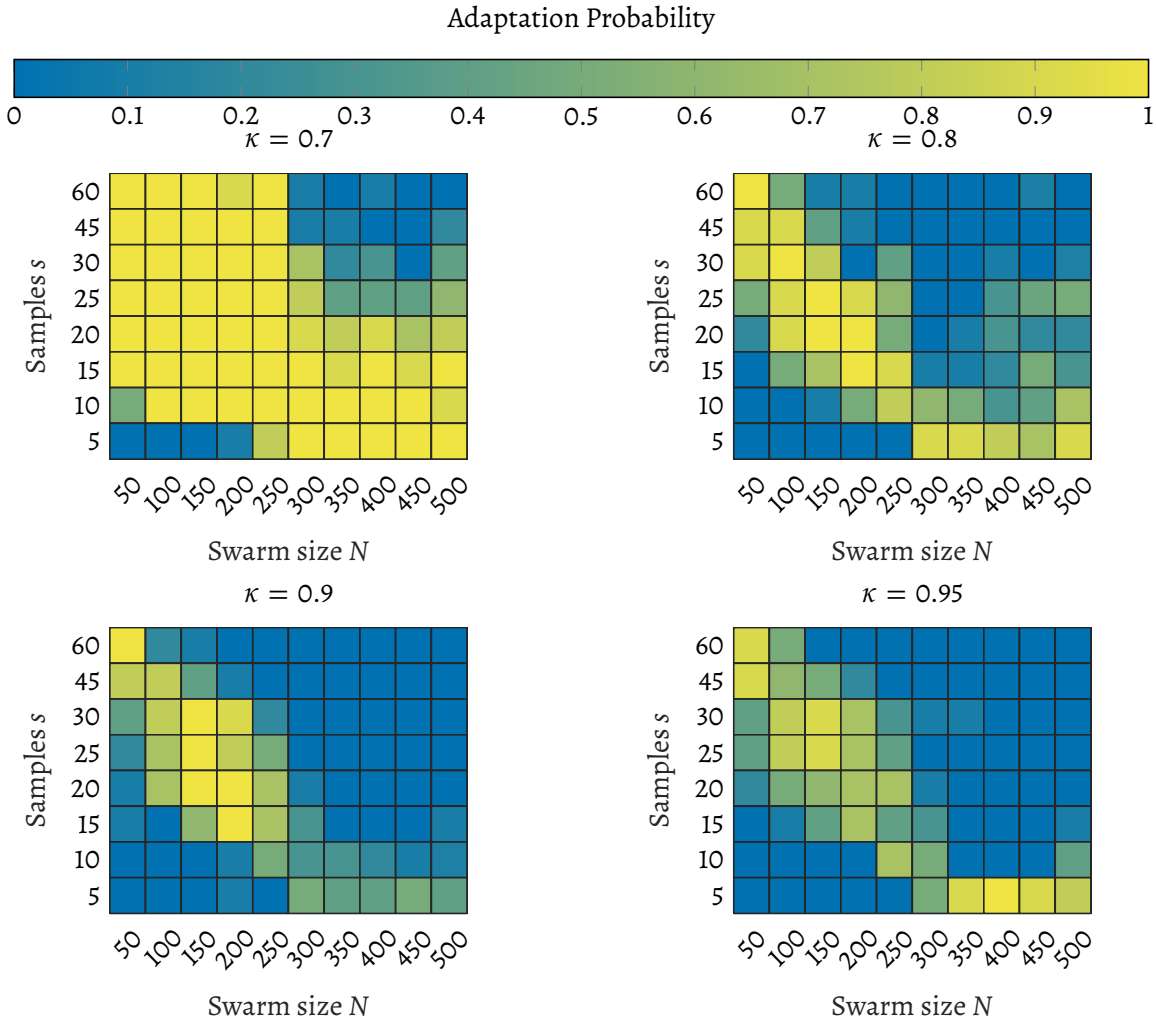


Figure 5.7: Fourth adaptation experiment with varied swarm size N and different number of samples s, with fixed time between samples $\delta_s = 1$ s and fixed communication range $r_c = 3$ cells ≈ 12.5 cm for different problem difficulties κ. Each data point is averaged over 10 independent simulation runs.

adapt for every parameter set.

Further, with this experiment we generalise the *less is more* effect (Talamali et al., 2021) to the new problem instance of collective perception. The *less is more* effect can be observed in the upper parts of Fig 5.3, where s is large and thus τ_s is large, because it provides the necessary condition of recruitment taking time. This effect causes the swarm to not adapt to the predominant colour for large communication ranges, as the large majority, committed to blue, is able to repeatedly mute minorities that make temporary discoveries of alternative options, i.e., colour yellow, further described in Section 2.1. The *less is more* effect holds well for communication ranges up to $r_c = 25$ cells. For larger communication ranges ($r_c > 25$ cells) the probability for adaptation rises again, see Fig. 5.3, top middle to top right corner. The observed effect of better adaptation for increasing communication range can be explained by the instability of the swarm's commitment due to single robots broadcasting their opinion to all robots, while a large majority is silent due

to sampling. The gradually increasing speaking extension helps to dampen this effect, but cannot completely mitigate it.
In the lower right part of the subfigures in Fig. 5.3, the swarm is able to adapt with low sampling time and high connectivity, caused by the large communication range. This can be explained by low sampling times enabling positive feedback cascades, which allow well connected swarms to react fast to environmental changes.
In the bottom left parts of the subfigures in Fig. 5.3, where the communication range is low and the sampling time is short, the robot swarm is not able to adapt. This is due to many switches, see Fig. 5.4, which indicates that the swarm detaches from the less concentrated colour (blue) but is not able to form a stable consensus in favour of the predominant colour yellow. The robot swarm fails to establish a stable majority, because the robots switch more often due to self-sourced information (noisy due to local perception) than due to social information (small communication range and thus sparse social interactions).
While reproducing the results of Talamali et al. (2021)'s work, we observed the interesting *slower is faster* effect, described in Section 2.1, which can be seen in the left part of the subfigures in Fig. 5.3. The robot swarm is able to adapt faster (during the experiment duration of 40 minutes) if the individual robot performs its task of estimating the concentration of a given colour at a slower pace. This effect occurs, when the robot connectivity is low, i.e., sporadic social interactions, due to a small communication range r_c. The finding of the *slower is faster* effect in our experiment aligns with the findings of Stark, Tessone, and Schweitzer (2008a) and Stark, Tessone, and Schweitzer (2008b), as they also found that increasing the switching probability, in our case a shorter sampling time leading to more self-sourced switches, slows the process of converging to a solution.
In the first experiment, we varied the sampling time τ_s by increasing the number of samples. In a second experiment, for confirming this statement we did an experiment, where we varied the time between samples δ_s over different communication ranges r_c, with a fixed number of samples $s = 15$. As mentioned in the previous section, we chose to continue with the problem difficulty of $\kappa = 0.9$, as it shows the dynamics most clearly. The dynamics of the robot swarm is similar to the first experiment, thus both parameters s and δ_s can be used for increasing the sampling time τ_s. However, in both cases, when slowing down the sampling process (increasing the sampling time), we also reduce the estimation noise, as we have more readings or the readings are less correlated to each other.
Therefore, we conducted our third experiment, where we test different levels of noise, see Fig. 5.6. This experiment investigates whether adaptation of sparse swarms is limited by high noise or quick recruitment. Sampling from the ground truth distribution allows us to disentangle the noise from sampling time and study their impact separately. In the case of no noise, see Fig. 5.6 $\sigma = 0$, the robot swarm is able to adapt for all cases, but large communication ranges and long sampling time. This indicates that the sampling time has no impact on the collective ability to adapt, because the robots have access to the ground truth, nullifying commitment switches towards the less concentrated colour based on self-sourced information. This allows the robots to adapt to the predominant colour, especially in the case of sporadic social interaction, i.e., small communication ranges. For larger communication ranges, i.e., richer social interaction, we observe the

less is more effect, where for longer sampling time, the majority of robots committed to the less concentrated colour overrule the robots discovering other options (yellow). In the case of noise, see Fig. 5.6 $\sigma = \{0.1, 0.2\}$, the swarm shows similar dynamics for adaptation as in the first and second experiment, that is adaptation either needs more time and a small communication range or recruitment takes less time and the communication range is larger. For less noise the swarm is able to adapt with a wider range of parameters. This experiment indicates that the requirements for the *slower is faster* effect are local communication and a noisy system, which are the typical conditions of swarm robotic systems. Further, with the third experiment, we can prove that the ability of adaptation is indeed due to the longer recruitment time and not due to the fact that longer sampling times also reduce the noise.
In the final experiment, we examine the link between communication range and swarm density. Therefore, we run experiments where we fix the communication range to $r_c = 3$ cells and vary the swarm size $N = [50, 500]$. Even though we only have few repetitions per data point we observe a clear trend. Also, because we have a grid of parameter sets and the curves are smoothly we can argue that the data are valid from a qualitative point of view. We test the ability of adaptation for different problem difficulties $\kappa = [0.7, 0.95]$. Similar to the first experiment, for simpler problems the robot swarm is able to adapt with a wider range of parameters and the robot swarm adapts for either large communication range and short sampling time or small communication range and large sampling time. The reasoning is similar to the first experiment. Further, we can confirm that the key parameter is connectivity of the robot swarm, which can either be increased by larger communication range, or by increasing the density of the robot swarm. The communication range is often a parameter defined by the hardware, or trades-off with other undesirable costs, such as heavily increased energy consumption (Rausch et al., 2019). On the other hand increasing the swarm density translates to buying more robots, assuming deploying the system in the same environment.

5.3 Symmetry Breaking

In this section we test the ability of symmetry breaking of the described robot behaviour in Section 4.2. We begin by describing the experimental set-up, followed by showing the results of the conducted experiments. Finally, we discuss the results.

Experimental Set-up

For testing the ability of symmetry breaking, we propose a symmetric best-of-n problem as described in Section 4.1. We vary the difficulty of the task by choosing different numbers of options $n = \{2, 3, 4, 5\}$.
In the beginning the robot swarm is split into n equal sized subpopulations, each committed to one colour. Thereby, the initial estimated quality of each robot is $\hat{\rho}_m = \frac{1}{n}$ for its committed option. We randomly initialise all the counters and the option the robot starts sampling to have a realistic state of the robot swarm in the beginning of the experiment.
In the experiments for adaptation we analysed the impact of the key parameters, sam-

pling time τ_s and connectivity of the robot swarm. These are also the key parameters for the symmetry breaking experiments. We vary the communication range $r_c = [1, 45]$ cells over different number of samples s. Thereby, we fix the time between samples to $\delta_s = 1$ s and the swarm size to $N = 50$.
For testing the second key parameter, connectivity of the swarm, we vary the swarm size $N = [50, 500]$ and keep the communication range fixed $r_c = 3$ cells.
Each experimental run is simulated for 40 minutes (76800 simulation steps, with 32 ms per simulation step).

Results

In the first experiment, we varied the sampling time by choosing different number of samples $s = [5, 60]$, and varied the connectivity of the robot swarm by using different static communication ranges $r_c = [1, 45]$ cells. We fixed the time between samples to

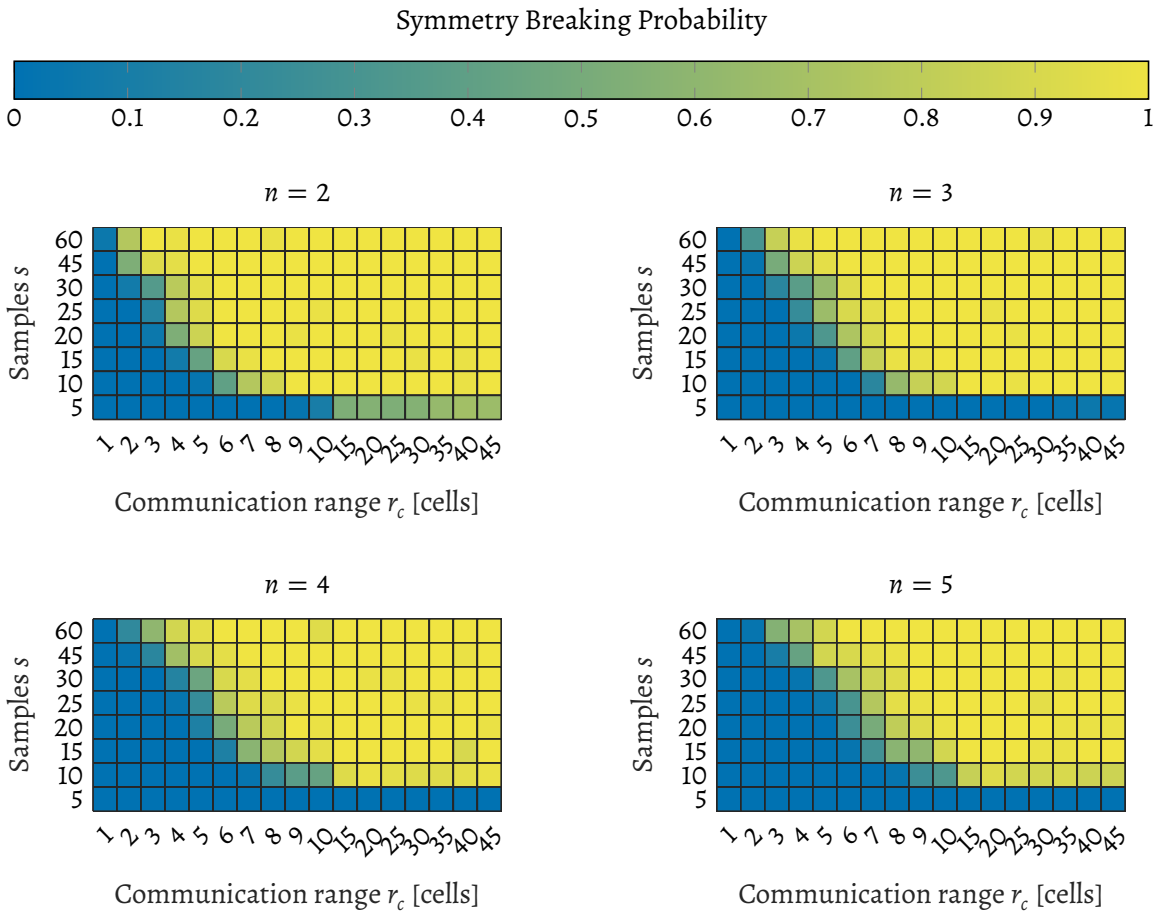


Figure 5.8: First symmetry breaking experiment with different number of equally good options n, with varied number of samples s over different static communication ranges r_c with fixed time between samples $\delta_s = 1$ s. The swarm size is $N = 50$ robots. For each data point we did 40 independent simulation runs.

$\delta_s = 1$ s and choose the swarm size $N = 50$ robots. The results for different number of colours can be seen in Fig. 5.8. We observe that the robot swarm is able to break the symmetry, if the communication range is large enough. Further, for more options, i.e., when the problem becomes more difficult, the robots need larger communication ranges or a longer sampling time for keeping the ability of symmetry breaking. In Fig. 5.9, we

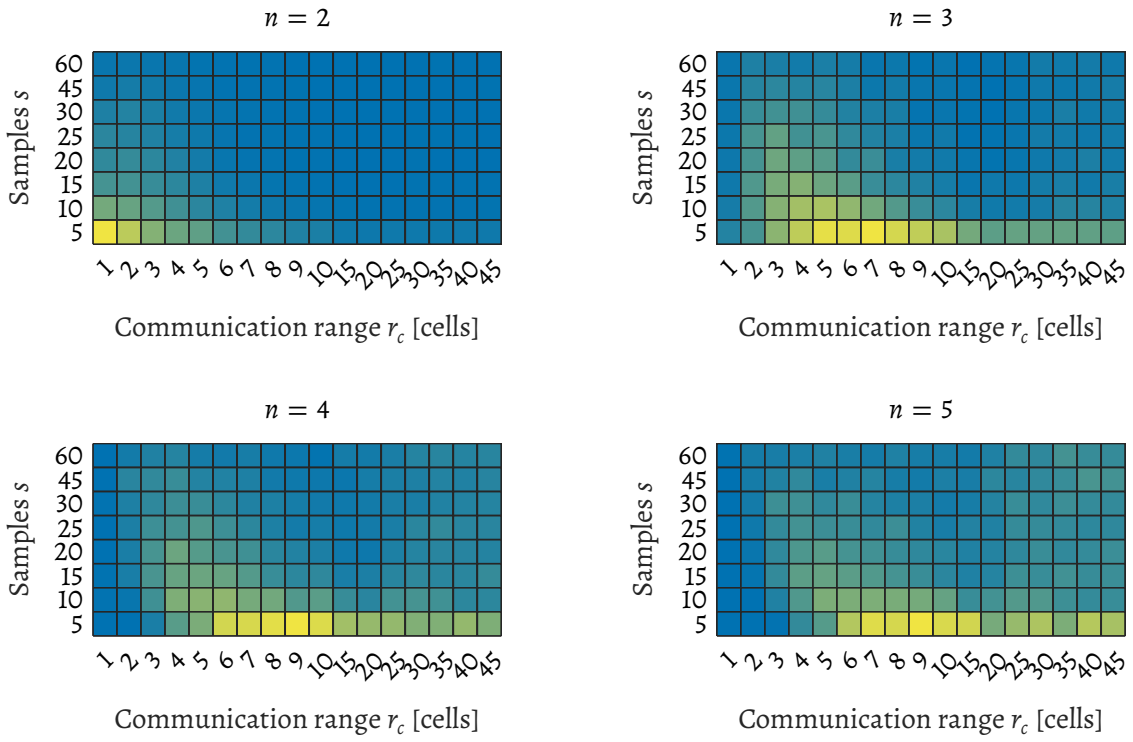


Figure 5.9: Average number of switches in the first symmetry breaking experiment, with varied number of samples $s = [5, 60]$ over different static communication ranges $r_c = [1, 45]$ cells. The switches follow different scales: $n = 2$ the maximum number of switches (yellow) are 50, $n = 3$ the maximum number of switches are 7, $n = 4$ the maximum number of switches are 5 and $n = 5$ the maximum number of switches are 5. We conducted 40 independent simulation runs per data point.

present the average switches of the first symmetry breaking experiment. It shows that the most switches happen if the sampling time τ_s is low, i.e., $s = 5$ samples.

In the second experiment we vary the swarm size $N = [50, 500]$ over the number of samples $s = [5, 60]$ for different number of colours $n = [2, 5]$. The time between samples is fixed to $\delta_s = 1$ s and the communication range is fixed to $r_c = 3$ cells. Similar to the first symmetry breaking experiment, the robot swarm is able to break the symmetry if there is enough social interaction, now caused by increased density. Further, increased sample time is also beneficial for the ability of symmetry breaking. The swarm is able to break the symmetry for a wider range of parameters, if the problem is easier. This experiment confronted us with the same issues as the density experiment of the adaptation study.

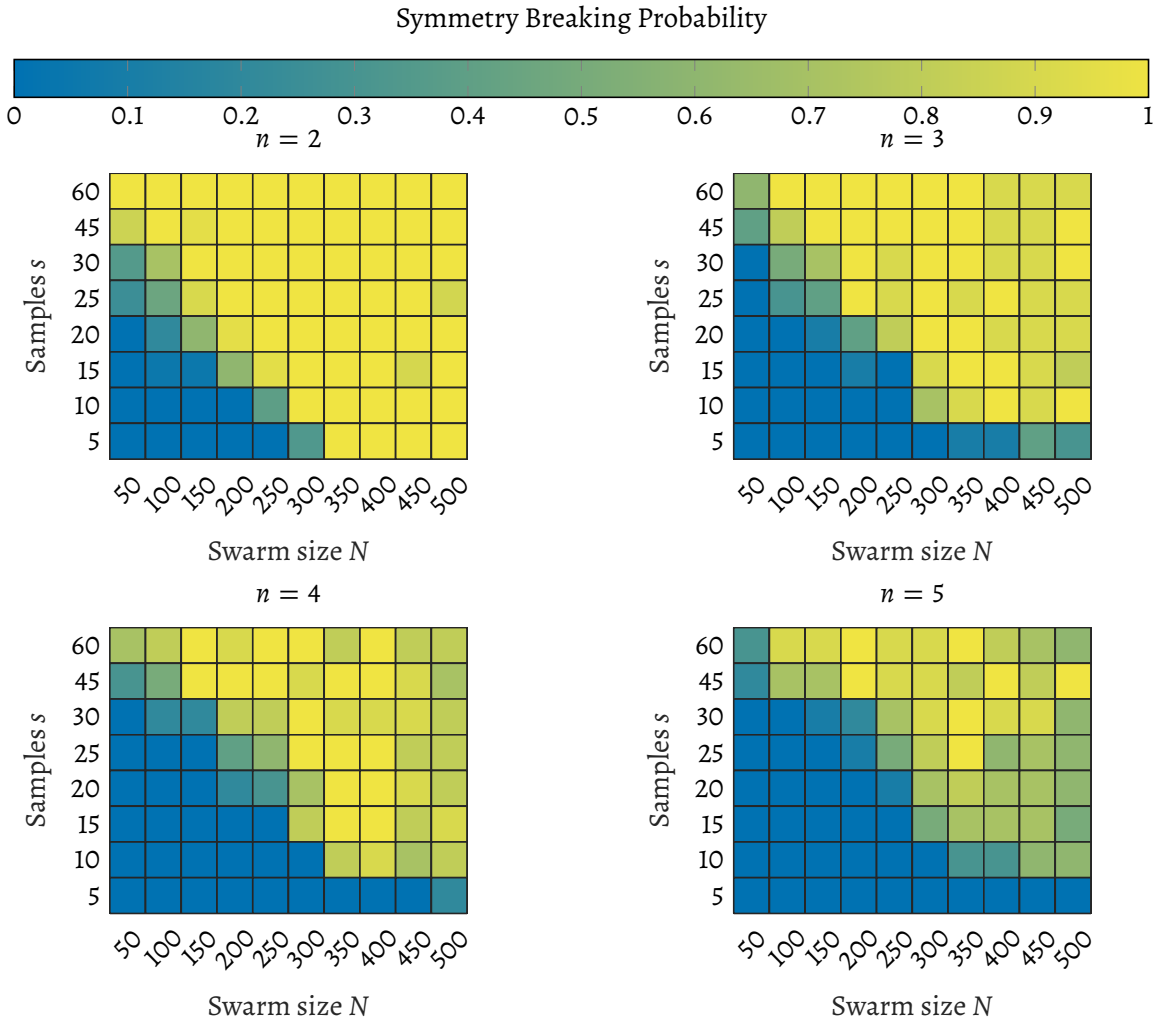


Figure 5.10: Second symmetry breaking experiment with different number of equally good options n, with varied number of samples s needed for making an estimate over different swarm sizes N with fixed time between samples $\delta_s = 1$ s and fixed communication range $r_c = 3$ cells.

Therefore, again we only can provide 10 independent simulation runs pre data point.

Discussion

In the aforementioned experiments, we study the robot swarm's ability of symmetry breaking. Therefore, we varied the key parameters of recruitment time, modelled by τ_s as the robots can only recruit robots after making their own estimate, and inter swarm connectivity, modelled by the communication range r_c in the first experiment and by increasing the swarm density in the second experiment.
For good connectivity of the swarm, i.e., Fig. 5.8 the right part in the subfigures, positive feedback cascades allow the robot swarm to break the symmetry. Further, the ability of symmetry breaking is influenced by the recruitment time, i.e., the time the robots need to sample an option and are "theoretically" uncommitted. For larger recruitment time,

i.e., the individual robot needs to take more samples for making an estimate, the swarm is more likely to break the symmetry, even for smaller communication ranges. On the other hand, if the recruitment time is shorter, i.e., fewer samples are required for making an estimate, the robot swarm loses the ability of symmetry breaking. This can be explained by the trade-off between updating the commitment through social or self-sourced information. For sporadic social interactions, when the communication range is low, the robots more often update their commitment through individual evidence. This effect gets amplified when the sampling time is low, due to only requiring few samples. This results in a state where the subpopulations of the robot swarm fluctuate because no subpopulation is able to recruit a large majority of robots in favour of their option, due to only sporadic social interaction. This fluctuating subpopulations also explain the switches presented in Fig. 5.9, where when the swarm is not able to break the symmetry the average number of switches goes up. This reasoning can be extended to the case of more social interaction between the robots. By increasing the social interactions it is easier for subpopulations of the robot swarm to disseminate their opinion, and by chance and through fluctuations and noise some subpopulation becomes the majority and through positive feedback convinces the whole swarm.
We notice, that the robot swarm is not able to break the symmetry for short sampling time, i.e., $s = 5$ and thus for short recruitment time. The number of switches indicate that the robot swarm is in some unstable state, because the robot swarm switches commitment often. Due to the very short recruitment time, switches happen relatively fast and instantaneous thus the swarm is not able to form a stable consensus on any option (stay committed to one option for more than 5 minutes).
In the second experiment, we varied the second key parameter by changing the swarm density instead of the communication range. Although, we did not perform many repetitions per data point, only 10 independent simulation runs, we have a grid of parameters and smooth dynamics, thus we can argue that the results are sufficient for a qualitatively analysis. The dynamics of the robot swarm are similar to when varying the communication range. This is due to the aforementioned trade-off of social and self-sourced evidence used for commitment update.

In both scenarios, adaptive best-of-n and symmetry breaking, we vary the robot swarm's connectivity by either increasing the communication range or increasing the swarm's density. In the following, we compare the influence of increasing the communication range and increasing the swarm's density on the robot swarm's connectivity.
For the varying communication range experiments, the swarm density is fixed to

$$d = 25\frac{\text{Robots}}{m^2} = \frac{50\ \text{Robots}}{2\ m^2}. \tag{5.2}$$

The robots do not know from how far they receive a message, i.e., the communication range is only important for sending messages. Thus, we only need to consider the sending robots' communication ranges in the following analysis. We can approximate the area the robot can send messages to using its communication range in cells

$$A_c = \pi r_c^2 = \pi(2.5 + 5(c-1))^2\ cm^2. \tag{5.3}$$

We can combine Eq. (5.2) and Eq. (5.3) to give a theoretical approximation of the average neighbourhood size depending on the communication range in cells

$$neigh(c) = dA_c = 25\frac{\pi(5c - 2.5)^2}{10000}. \tag{5.4}$$

We acknowledge that this simple model is only a rough theoretical approximation, which is limited by the fact that the Kilogrid is only $1\,m \times 2\,m$, thus a communication range for 45 cells which we have chosen so that the robots definitely speak over the whole environment does not account the limited space. In Fig. 5.11 we plot the theoretical average

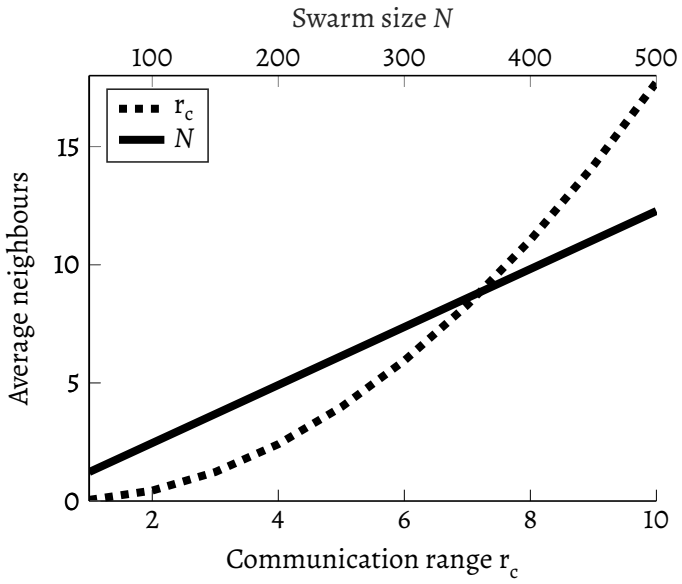


Figure 5.11: Density calculation.

number of neighbours per robot. Next, we can compare the average number of neighbours from the first and second symmetry breaking experiment, see Fig. 5.8 and Fig. 5.10 respectively. For the case of $r_c = 1$ the average number of neighbours is $neigh(1) \approx 0$, that is why the robot swarm is never able to break the symmetry. When the recruitment time is short, i.e., only $s = 5$ samples are required to make an estimate the swarm is unstable and cannot establish a stable majority (5 minutes).
We compare the frontier at which the robot swarm starts to break the symmetry. We do so by checking for the problem instance $n = 2$. The frontier starts at $s = 30$ and $r_c = 3$ or $N = 50$. The behaviour is similar because it is a identical parameter set. Then the frontier goes down approximately linear to the point $s = 5$ and $r_c = 7$, $N = 300$ respectively. These points have a similar average number of neighbours. Comparing the average number of neighbours based on communication range or density, see Fig. 5.11, we see that the frontier depends on the average neighbours. Similar dynamics can be found for the other problem instances $n = \{3, 4, 5\}$.
Because of the polynomial increase in the average number of neighbours for increasing communication range, the swarm can break the symmetry with high reliability.
A communication range between 8 and 9 cells represents the density of 500 robots. Sim-

ilar to the adaptation study the user has the design choice of increasing density or increasing communication range.

6

Dynamic Communication Range

In the previous chapter we tested our robot behaviour for the dynamic best-of-*n* scenario and the symmetry breaking scenario. Thereby, we used static communication ranges for the individual robots. The experiments revealed that the robot swarm is able to

- adapt for smaller static communication ranges (*less is more* effect) and,
- break the symmetry for larger static communication ranges.

For real world scenarios, where the robot swarm has to solve both problems simultaneously, the problem of choosing the right communication range arises. Our previous experiments indicate that choosing the right communication range can be difficult or even infeasible. To solve the problem of choosing the right communication range, we propose to dynamically adjust the communication range of the individual robots. Thereby, the decisions on the adjustment should be decentralised and autonomous.

We structure this chapter as follows, first, we describe the extension of the robot behaviour and how to change the robot controller to integrate the new feature of individually adjusting the communication range. Afterwards, we present the experimental setup, the obtained results and the discussion.

6.1 Behaviour Extension

We extend the current robot behaviour by allowing the robot to choose its communication range individually. In the following, we present the implementation of this extension. As foreshadowed in Section 4.3, checking for adjusting the communication range is done every control loop cycle, see Fig. 4.5 update communication range.
The robot increases its communication range after self-sourced commitment updates or quality updates. This is the case after the transitions of arrows A, D and E in Fig. 4.3. Increasing communication range solely based on self-sourced information helps to not spread misinformation received from other (malfunction, malicious) robots. Another, more intuitive explanation is that if a robot discovers a new (better) option it wants to disseminate this information. The communication range is lowered, if either the threshold τ_1 is reached or the robot changes opinion based on social information, arrows B or

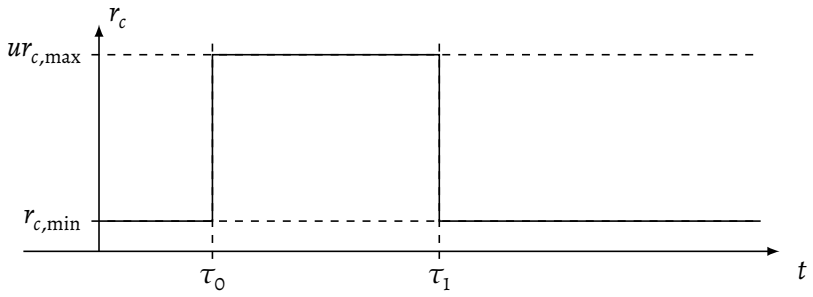


Figure 6.1: Step function for increasing communication range. It shows how the communication range changes over time, assuming that at τ_0 the robot switches or updates commitment based on self-sourced information.

C in Fig. 4.3.

To be in line with our design concept we opted for a minimalistic approach, thus we implement the dynamic communication range update based on a step function, see Fig. 6.1. We chose the step function, because it requires a minimal set of parameters and minimal computational effort. These parameters include the choice of the communication range and the time the robot should use the large communication range.
We need to choose the minimal and maximal communication range of the robot such that using the minimal communication range the robot swarm is able to adapt and using the maximal communication range the robot swarm is able to break the symmetry. Based on the previous experiments from Chapter 5, we choose the minimal communication range $r_{c,\min} = 3$ cells and the maximal communication range $r_{c,\max} = 45$ cells.
The robot selects its communication range between the minimal and maximal communication range depending on the quality estimation improvement u. We calculate u as

$$u = 10(\hat{\rho}_d - \hat{\rho}_m), \tag{6.1}$$

with $\hat{\rho}_d$ the newly discovered concentration estimation and $\hat{\rho}_m$ the robot's current concentration estimation. We scale the difference by 10, because it can be very small, e.g., in the case of $\kappa = 0.9$ the ground truth difference is 0.05. Thus, the increased communication range is given by $ur_{c,\max}$.
The last parameter to choose is the threshold τ_1 for how long the communication range should be increased. We set τ_1 to 30 s, because it has been empirically shown to work best.

6.2 Experiments

After introducing the implementation of dynamically choosing the communication range, we present the conducted experiments. Besides testing the new extension on the previous scenarios: dynamic best-of-n and symmetry breaking, we also introduce a new scenario, that is a combination of both. In the new scenario, we propose an environment

with three colours, where one colour is less concentrated than the other two colours, which have equal concentration. Therefore, we first extend our metrics of adaptation for the new environment. Afterwards, we present the experimental set-up and the results of the conducted experiments, which we discuss thereafter.

Extension of Metrics

For the new environment with three options, we need to adjust the metric of adaptation by combining our previous definition with the definition of symmetry breaking.
Therefore, we say that if there are multiple colours with the highest concentration the robot swarm adapts when a large majority commits to either of these colours. The majority is 70% of the swarm and we require it to stay committed to one colour for at least 5 minutes, to not count short lived fluctuations as adapted.
An overview of the cell concentration for different problem difficulties κ of the new environment is given in Tab. 6.2.

Table 6.2: Overview of the cell concentration for different κ for the adaptive symmetry breaking scenario. The total number of cells is 684.

κ	# Cells / ρ_l	# Cells / ρ_h
0.7	178 / 0.26	253 / 0.37
0.8	196 / 0.28	244 / 0.36
0.9	212 / 0.30	236 / 0.35
0.95	220 / 0.32	232 / 0.34

Experimental set-up

First, we test the extension of dynamically choosing communication range on the previous presented scenarios of dynamic best-of-n and symmetry breaking. Therefore, we varied the key parameter of sampling time by choosing different number of samples s and keeping the time between samples $\delta_s = 1$ s fixed. For the dynamic best-of-n scenario we choose a problem difficulty of $\kappa = 0.9$. For the symmetry breaking scenario we choose $n = 4$ options.
Next, we propose a new environment which combines both scenarios. Therefore, we consider an environment with $n = 3$ options. Without loss of generality, we assume $\rho_1 < \rho_2 = \rho_3$. Similar to the dynamic best-of-n scenario, we assume a instantaneous change of concentration in the beginning of the experiment, which is the reason why the robot swarm starts fully committed to the less concentrated colour with a high quality estimate of $\hat{\rho}_m = 0.8$. For a realistic state of the robot swarm we initialise all timers and the colour the robots sample randomly. This set-up allows us to study both scenarios, adaptation and symmetry breaking, at the same time, because first the swarm needs to adapt, abandon option o_1 (blue), and second break the symmetry, commits to either option o_2 (yellow) or option o_3 (red).

We conduct experiments, where we vary the key parameter sampling time by changing the number of samples and by changing the time between samples. Similar to the best-of-n scenario, we perform the experiments for different problem difficulties $\kappa \in \{0.7, 0.8, 0.9, 0.95\}$.
In all experiments we include the results for the static communication ranges $r_c = 3$ cells and $r_c = 45$ cells for providing a baseline how local and global communication performs in the scenarios compared to our new extension of the robot controller. All experiments have a swarm size of $N = 50$ robots and are simulated for 40 minutes (76800 simulation steps, with 32 ms per simulation step).

Results

In the first experiment, we test the extension of dynamic communication range in the dynamic best-of-n scenario. The results for problem difficulty $\kappa = 0.9$ can be seen in

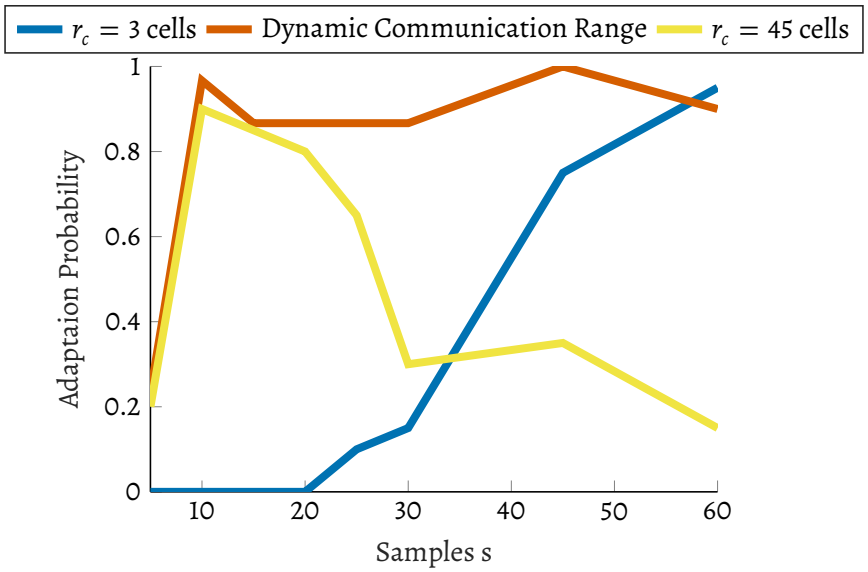


Figure 6.3: Experiment for dynamic communication range in the dynamic best-of-n scenario with problem difficulty $\kappa = 0.9$. We vary the number of samples and keep $\delta_s = 1$ s fixed. For comparison we added the probabilities for the static communication ranges $r_c = 3$ cells and $r_c = 45$ cells. The swarm size is $N = 50$ robots and we conducted 30 independent simulation runs per data point.

Fig. 6.3. We added the probabilities of static local and global communication for comparison. The dynamic communication range is able to adapt with a high probability for $s \geq 10$.

In a second experiment, we tested the extension of dynamic communication range in the symmetry breaking scenario. The results for $n = 4$ options can be seen in Fig. 6.4. For comparison we added the probabilities of static local and global communication. The dynamic communication range performs similar to the static global communication range.

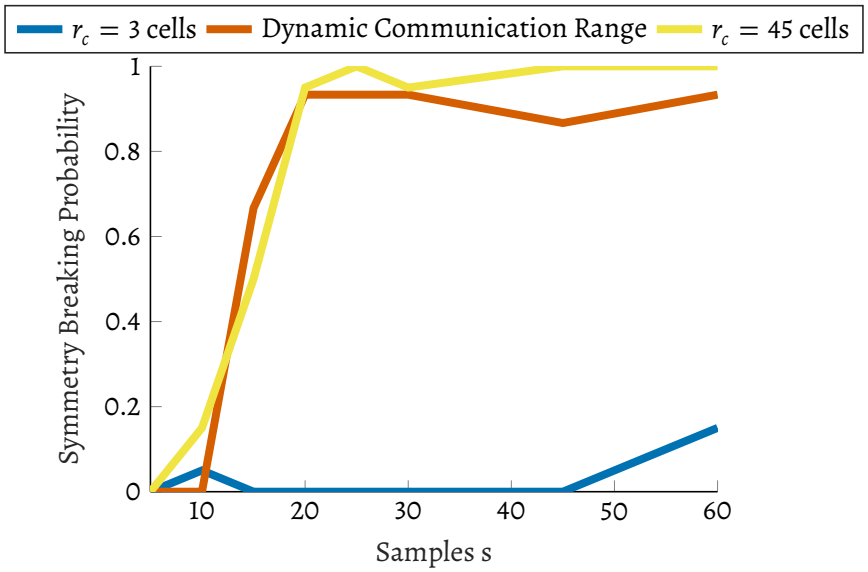


Figure 6.4: Experiment for dynamic communication range in the symmetry breaking scenario for $n = 4$ options. We vary the number of samples and keep $\delta_s = 1$ s fixed. For comparison we added the probabilities for the static communication ranges $r_c = 3$ cells and $r_c = 45$ cells. The swarm size is $N = 50$ robots and we conducted 30 independent simulation runs per data point.

In the third experiment, we test the new scenario adaptive symmetry breaking. For providing a baseline, we test the static local communication range. Results for $r_c = 3$ cells and different problem difficulties can be seen in Fig. 6.5. The robot swarm is able to adapt for certain parameter sets, which are larger if the problem difficulty is simpler.

Next, we test the static global communication range in the new environment by choosing a communication range of $r_c = 45$ cells. The results for different problem difficulties can be seen in Fig 6.6. The robot swarm is able to adapt for a wide range of parameters, which decreases when the problem becomes more difficult.

Finally, we test the extension of dynamic communication in the new environment. Results for different problem difficulties can be seen in Fig. 6.7. The robot swarm is able to adapt if the sampling time is long enough.

Discussion

In the first experiment, we tested the new extension of dynamic communication range in the dynamic best-of-n scenario. For comparison we included static local and global communication ranges. We observe that the new extension outperforms both static communication ranges and is able to adapt with a high probability for all sample times greater than 10 s.
In the second experiment, we tested the new extension in the symmetry breaking scenario. Again, for comparison we included static local and global communication range.

We observe that the dynamic communication range extension has a similar performance as the static global communication range, that is high symmetry breaking probability for sampling times greater than 20 s.
These experiments show, that letting the robots individually choose their communication range increases the performance in case of adaptation, and shows similar (best case) performance in the case of symmetry breaking. Further, it shows that only minimalistic rules are required for the performance gains.

In a third set of experiments, we introduce a new environment with three colours, where one colour is less concentrated than the other two colours, which are equally concentrated. The new environment allows us to study the swarm's ability of simultaneously adapting and breaking the symmetry.
Therefore, we conducted two experiments to have a baseline on how the static local and global communication ranges perform in the new environment. The static local communication range was able to adapt for certain parameter sets, i.e., for certain sampling times. Interestingly, there is a lower and upper limit for the sampling time, see lower left or upper right corner of Fig. 6.5 respectively. For problem difficulty $\kappa = 0.95$ the robot swarm almost loses the ability of adaptation.
Similar to the static local communication range, the static global communication range was able to adapt for certain parameter sets, see Fig. 6.6. Thereby, the static global communication range has a wider parameter set.
However, both approaches are limited to certain parameter sets, i.e., the ability to adapt depends on the sampling time, which has a lower and upper limit. In the final experiment we tested the extension of dynamic communication range in the new environment. We observe that if the sampling time is long enough the swarm adapts with a high probability, see Fig. 6.7. The parameter range is not as dependent on the problem difficulty as on the static communication range experiments. Further, the dynamic communication range extension has only a lower limit. Having only a lower limit brings the advantage that if you do not know much about your problem, you can just choose a long sampling time to ensure that the robot swarm is able to adapt. Instead if the communication range is chosen statically it could be that you exceed the upper bounds and the swarm looses the ability of adaptation. Thus choosing a dynamic communication range eases the problem of finding the correct parameters.

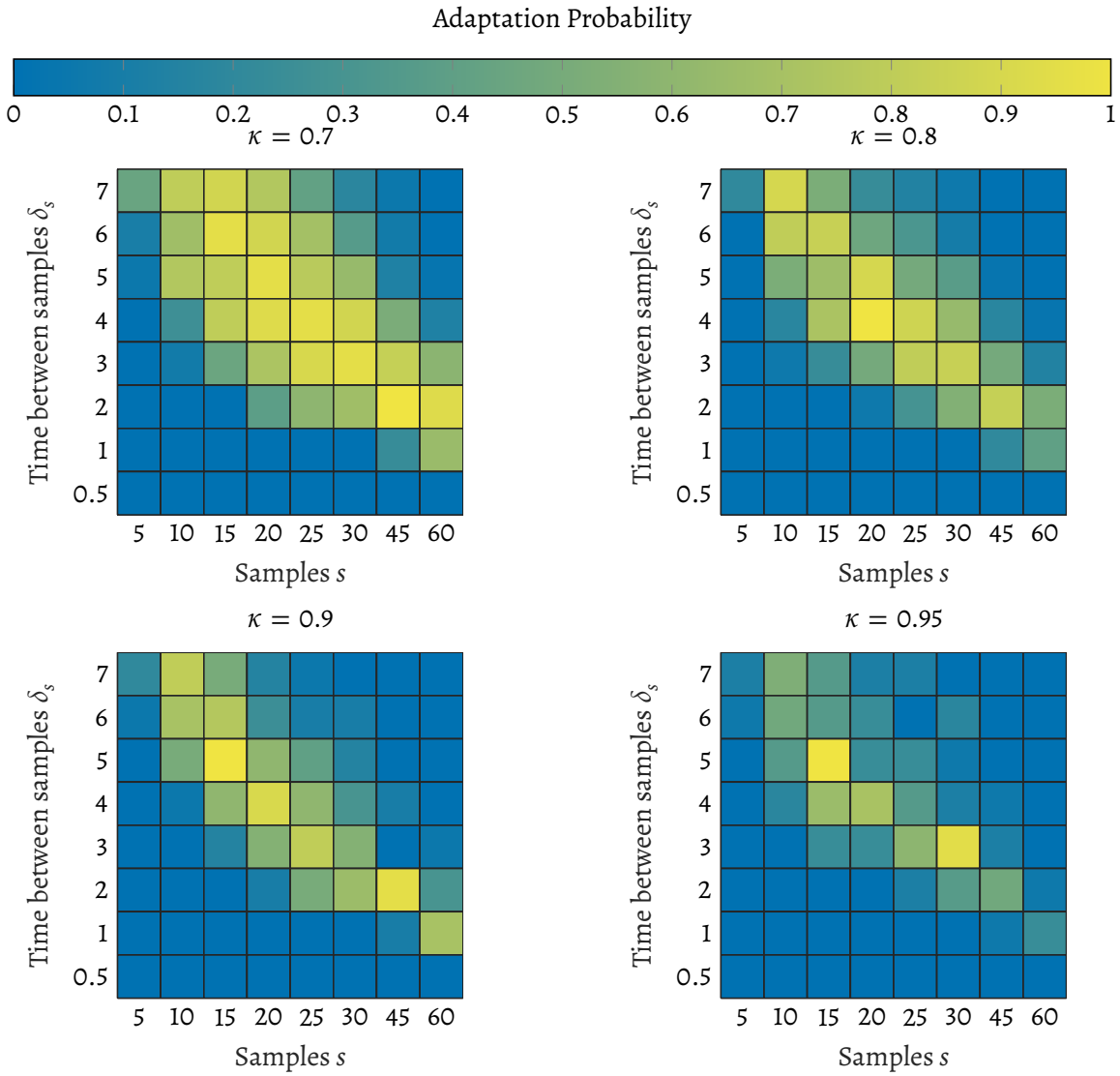


Figure 6.5: Results of the adaptive symmetry breaking experiment for different problem difficulties κ. We vary the sampling time by choosing different number of samples and different time between samples. The range is static and set to $r_c = 3$ cells. The swarm size is $N = 50$ robots and we conducted 30 independent simulation runs per data point.

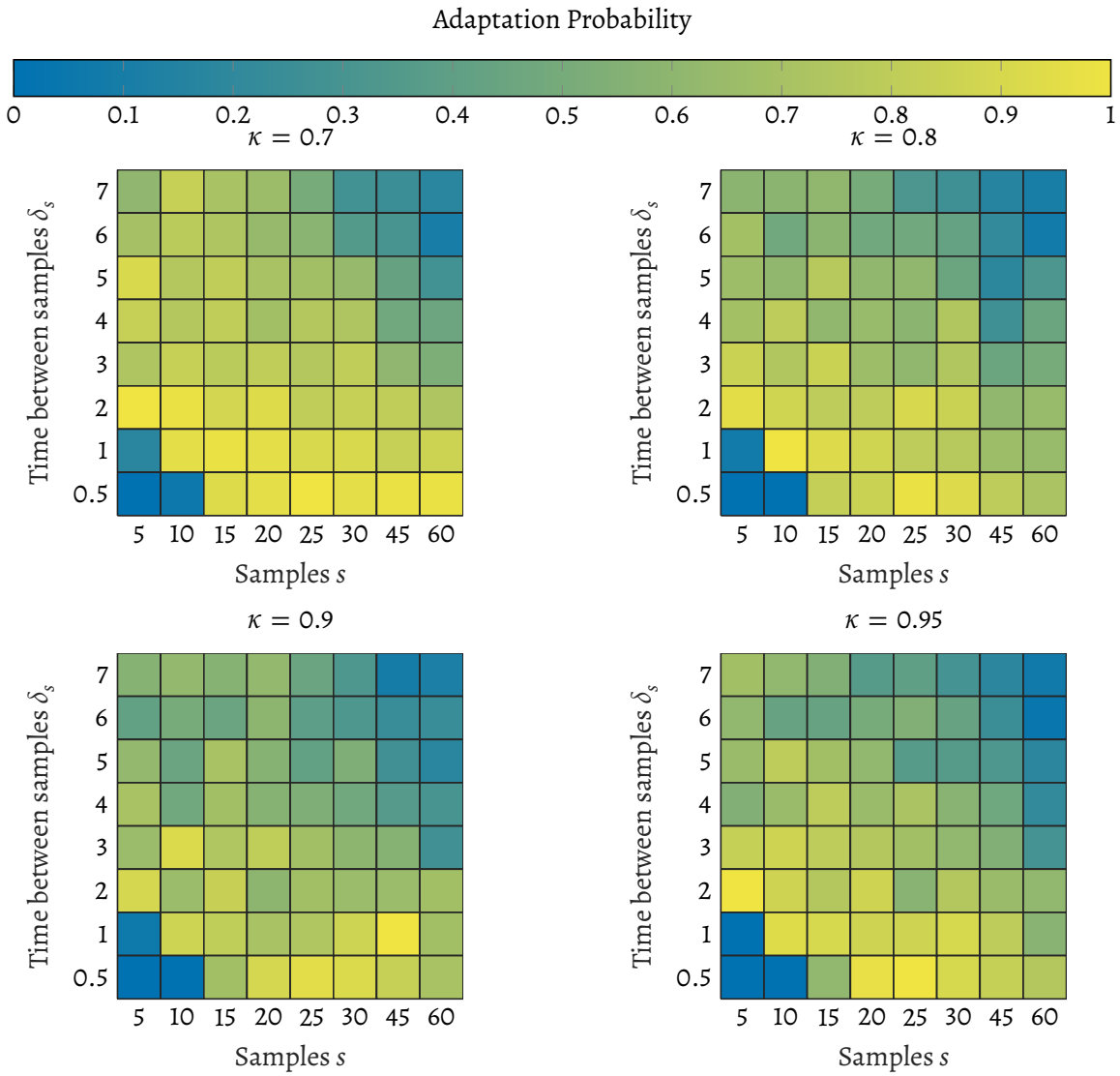


Figure 6.6: Results of the adaptive symmetry breaking experiment for different problem difficulties κ. We vary the sampling time by choosing different number of samples and different time between samples. The communication range is static and set to $r_c = 45$ cells. The swarm size is $N = 50$ robots and we conducted 30 independent simulation runs per data point.

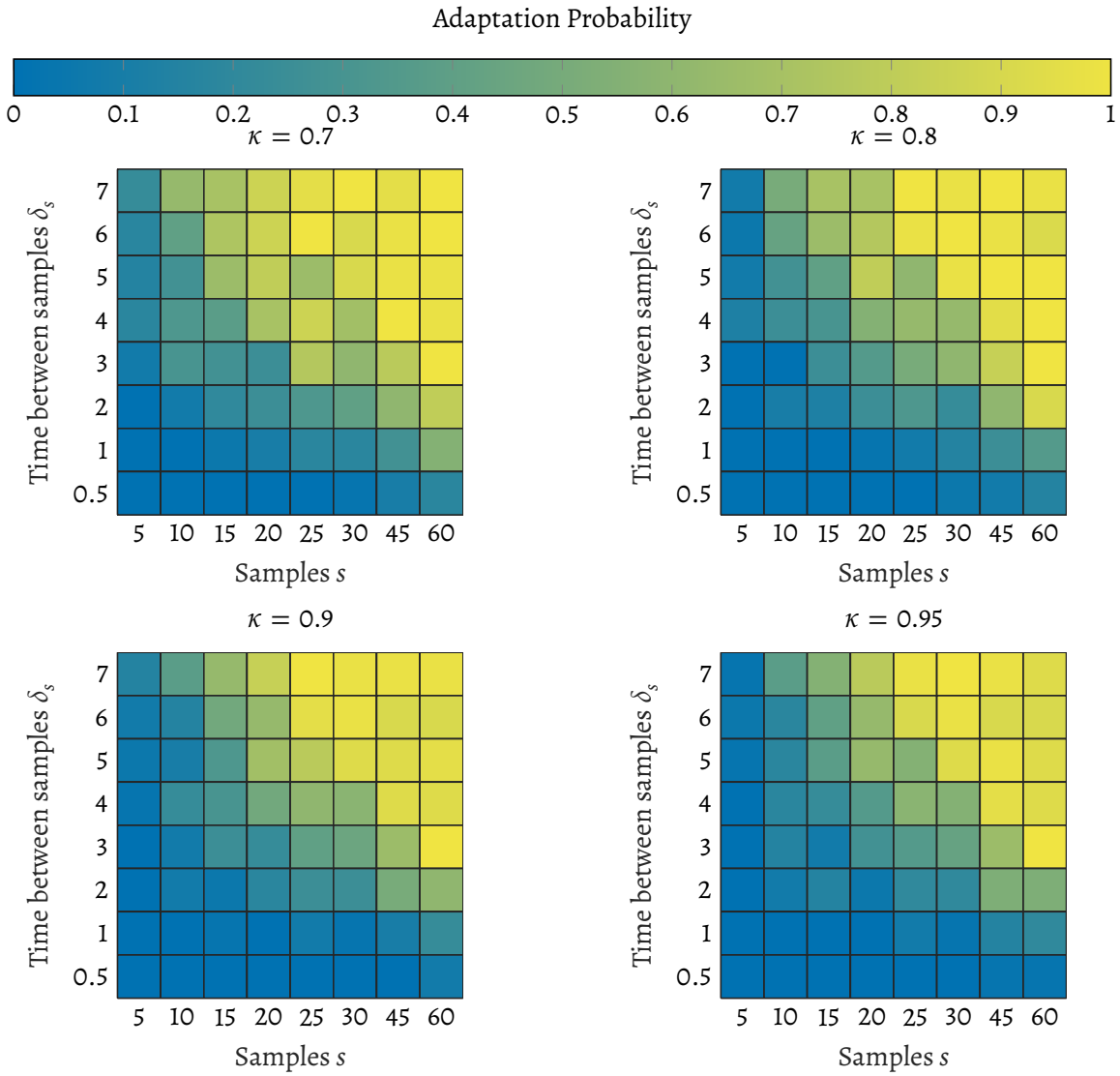


Figure 6.7: Results of the adaptive symmetry breaking experiment for different problem difficulties κ. The communication range is adjusting as presented earlier, with $r_{c,\min} = 3$ and $r_{c,\max} = 45$. We choose the threshold $\tau_1 = 30$ s. The swarm size is $N = 50$ robots and we conducted 30 independent simulation runs per data point.

7

Conclusion

Summary

In this thesis we studied the collective perception problem in three different scenarios: dynamic best-of-*n*, symmetry breaking and adaptive symmetry breaking. Therefore, we designed a minimalistic robot behaviour based on Talamali et al. (2021)'s work, which implements the social interaction pattern cross-inhibition. We extended the robot behaviour by the gradually increasing speaking extension, which yields more stability for the robot swarm.
For conducting our experiments we used the Kilogrid as augmented reality for the Kilobot. For the simulated experiments we developed an ARGoS 3 extension, which simulates the Kilogrid. Similar to the design concepts of ARGoS 3, we designed the extension such that it offers the same interfaces as the real Kilogrid control software.
While studying the robot swarm's ability to adapt in the dynamic best-of-*n* scenario, we were able to reproduce the *less is more* effect for the new problem of collective perception. The new experimental set-up, featuring the Kilogrid, allows us to precisely choose the sampling time of individual robots and the swarm's connectivity, which was not possible before. By varying the sampling time we observed a new effect: the *slower is faster* effect, that is the robot swarm is able to adapt (faster), if the individual robots do their task of sampling the environment slower. It occurs when the communication range is small, i.e., the robots have only few social interaction.
In the second scenario we study the robot swarm's ability of symmetry breaking. Thereby, we found that the robot swarm is able to break the symmetry if there is rich social interactions, i.e., if the communication range is large or the swarm density is high.
These findings contradict with the swarm's ability of adaptation, as described by the *less is more* effect, the robot swarm adapts only for small communication ranges. Combining the problems of adaptation and symmetry breaking, it becomes difficult or even infeasible to choose a communication range that allows the robot swarm to solve both problems simultaneously. That is why we propose another extension to the robot behaviour that is letting the robots choose their communication range individually, based on the current situation. The new extension was tested in all three scenarios: dynamic best-of-*n*, symmetry breaking and adaptive symmetry breaking. We were able to show, that dynamically adjusting the communication range of the robots is a promising solution, which can be

implemented following the minimalistic design concept. Further, it simplifies finding parameters for solving both problems in unknown environments.

Future Work

The next step is to deploy the experiments on the real Kilobots. While implementing the novel way of communication for the Kilobots we faced the issue that module to module communication is not properly supported by the Kilogrid control software. We are able to send messages between modules (direct addressing and broadcasting), but these messages block the control software, which iterates sequentially through all modules for collecting the tracking data. At some point, the module, which receives our user message and the tracking data request from the dispatcher, hangs up. The Kilogrid control software then waits for the hung up module to respond, with the result that the Kilogrid cannot track the Kilobots and we cannot send any control commands to the modules.
Although we cannot use the novel way of message transmission we can run experiments using direct robot to robot communication, i.e., local communication (approximately $2 < r_c < 3$). With this communication range we can show the *slower is faster* effect, as it only appears in local communication range. We already did some real world experiments[7] for proving transferability of the in simulation developed code. We were able to use the same code with slight modifications (increasing counters needed for the random walk, etc.).
Another issue we need to address is to modify the Kilobot plug-in for ARGoS 3, as it spawns a process for each simulated robot and thus makes large scale simulations problematic. Finally, we can extend our study by testing more different parameter sets, e.g., different thresholds τ_1. Further, we can study different social interaction patterns, e.g., the direct switching pattern, which was also studied by Talamali et al. (2021) or different communication range update behaviours, such as linear or exponential decrease of the communication range.

[7] https://doi.org/10.5281/zenodo.6611777

Bibliography

Antoun, A., Valentini, G., Hocquard, E., Wiandt, B., Trianni, V., and Dorigo, M. (2016). Kilogrid: A modular virtualization environment for the Kilobot robot. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3809–3814. DOI: 10.1109/IROS.2016.7759560.

Aust, T., Talamali, M.S., Dorigo, M., Hamann, H., and Reina, A. (2022). The Hidden Benefits of Limited Communication and Slow Sensing in Collective Monitoring of Dynamic Environments.

Bartashevich, P. and Mostaghim, S. (2021). Multi-featured collective perception with Evidence Theory: tackling spatial correlations. Swarm Intelligence *15*, 83–110. DOI: 10.1007/s11721-021-00192-8.

Beni, G. (2005). From swarm intelligence to swarm robotics. In International Workshop on Swarm Robotics, DOI: doi.org/10.1007/978-3-540-30552-1_1.

Bettstetter, C., Hartenstein, H., and Pérez-Costa, X. (2004). Stochastic Properties of the Random Waypoint Mobility Model. Wireless Networks *10*, 555–567. DOI: 10.1023/B:WINE.0000036458.88990.e5.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence *7*, 1–41. DOI: 10.1007/s11721-012-0075-2.

Dorigo, M. and Şahin, E. (2004). Guest Editorial. Autonomous robots *17*. DOI: 10.1023/B:AURO.0000034008.48988.2b.

Ebert, J.T., Gauci, M., and Nagpal, R. (2018). Multi-feature collective decision making in robot swarms. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1711–1719.

Ebert, J.T., Gauci, M., Mallmann-Trenn, F., and Nagpal, R. (2020). Bayes Bots: Collective Bayesian Decision-Making in Decentralized Robot Swarms. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 7186–7192. DOI: 10.1109/ICRA40945.2020.9196584.

Font Llenas, A., Talamali, M.S., Xu, X., Marshall, J.A.R., and Reina, A. (2018). Quality-Sensitive Foraging by a Robot Swarm Through Virtual Pheromone Trails. In Swarm In-

telligence, Cham: Springer International Publishing, pp. 135–149. DOI: 10.1007/978-3-030-00533-7_11.

Franks, N.R., Pratt, S.C., Mallon, E.B., Britton, N.F., and Sumpter, D.J.T. (2002). Information flow, opinion polling and collective intelligence in house–hunting social insects. Philosophical Transactions of the Royal Society of London. B: Biological Sciences *357*, 1567–1583. DOI: 10.1098/rstb.2002.1066.

Gauci, M., Chen, J., Li, W., Dodd, T.J., and Groß, R. (2014). Self-organized aggregation without computation. The International Journal of Robotics Research *33*, 1145–1161. DOI: 10.1177/0278364914525244.

Gershenson, C. and Helbing, D. (2015). When slower is faster. Complexity *21*, 9–15. DOI: 10.2139/ssrn.2639808.

Hamann, H. (2018). Swarm Robotics: A Formal Approach. 1st ed. Cham: Springer. ISBN: 978-3-319-74526-8. DOI: 10.1007/978-3-319-74528-2.

Holley, R.A. and Liggett, T.M. (1975). Ergodic Theorems for Weakly Interacting Infinite Systems and the Voter Model. The Annals of Probability *3*, 643–663. DOI: 10.1214/aop/1176996306.

Jafferis, N.T., Helbling, E.F., Karpelson, M., and Wood, R.J. (2019). Untethered flight of an insect-sized flapping-wing microscale aerial vehicle. Nature *570*, 491–495. DOI: 10.1038/s41586-019-1322-0.

Jeanson, R., Dussutour, A., and Fourcassié, V. (2012). Key Factors for the Emergence of Collective Decision in Invertebrates. Frontiers in Neuroscience *6*. DOI: 10.3389/fnins.2012.00121.

Minimalist Robot Swarms, A.S. for Team Formation in (2022). Luigi Feola and Vito Trianni. IEEE Robotics and Automation Letters *7*, 4079–4085. DOI: 10.1109/LRA.2022.3150479.

Özdemir, A., Gauci, M., Bonnet, S., and Groß, R. (2018). Finding Consensus Without Computation. IEEE Robotics and Automation Letters *3*, 1346–1353. DOI: 10.1109/LRA.2018.2795640.

Pais, D., Hogan, P.M., Schlegel, T., Franks, N.R., Leonard, N.E., and Marshall, J.A.R. (2013). A Mechanism for Value-Sensitive Decision-Making. PLOS ONE *8*, 1–9. DOI: 10.1371/journal.pone.0073216.

Parker, C.A.C. and Zhang, H. (2004). Biologically inspired decision making for collective robotic systems. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 1, pp. 375–380. DOI: 10.1109/IROS.2004.1389381.

Parker, C.A.C. and Zhang, H. (2009). Cooperative Decision-Making in Decentralized Multiple-Robot Systems: The Best-of-N Problem. IEEE/ASME Transactions on Mechatronics *14*, 240–251. DOI: 10.1109/TMECH.2009.2014370.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., et al. (2012). ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems. Swarm Intelligence *6*, 271–295. DOI: 10.1007/s11721-012-0072-5.

Pinciroli, C., Talamali, M.S., Reina, A., Marshall, J.A.R., and Trianni, V. (2018). Simulating Kilobots Within ARGoS: Models and Experimental Validation. In Swarm Intelligence, (Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Reina, A., and Trianni, V., eds.). Springer International Publishing, pp. 176–187. DOI: 10.1007/978-3-030-00533-7_14.

Prasetyo, J., De Masi, G., and Ferrante, E. (2019). Collective decision making in dynamic environments. Swarm Intelligence *13*, 217–243. DOI: 10.1007/s11721-019-00169-8.

Prasetyo, J., De Masi, G., Ranjan, P., and Ferrante, E. (2018). The best-of-n problem with dynamic site qualities: Achieving adaptability with stubborn individuals. Swarm Intelligence (ANTS 2018), vol. 11172. LNCS. Cham: Springer, pp. 239–251. DOI: doi.org/10.1007/978-3-030-00533-7_19.

Pratissoli, F., Reina, A., Lopes, Y.K., Sabattini, L., and Groß, R. (2019). A Soft-Bodied Modular Reconfigurable Robotic System Composed of Interconnected Kilobots. In 2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), pp. 50–52. DOI: 10.1109/MRS.2019.8901061.

Rausch, I., Reina, A., Simoens, P., and Khaluf, Y. (2019). Coherent collective behaviour emerging from decentralised balancing of social feedback and noise. Swarm Intelligence *13*, 321–345. DOI: 10.1007/s11721-019-00173-y.

Reina, A., Valentini, G., Fernández-Oto, C., Dorigo, M., and Trianni, V. (2015). A Design Pattern for Decentralised Decision Making. PLOS ONE *10*, 1–18. DOI: 10.1371/journal.pone.0140950.

Reina, A., Cope, A.J., Nikolaidis, E., Marshall, J.A.R., and Sabo, C. (2017a). ARK: Augmented Reality for Kilobots. IEEE Robotics and Automation Letters *2*, 1755–1761. DOI: 10.1109/LRA.2017.2700059.

Reina, A., Marshall, J.A.R., Trianni, V., and Bose, T. (2017b). Model of the best-of-*N* nest-site selection process in honeybees. Phys. Rev. E *95*, 052411–052426. DOI: 10.1103/PhysRevE.95.052411.

Rubenstein, M., Ahler, C., and Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. In 2012 IEEE International Conference on Robotics and Automation, pp. 3293–3298. DOI: 10.1109/ICRA.2012.6224638.

Rubenstein, M., Cornejo, A., and Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. Science *345*, 795–799. DOI: 10.1126/science.1254295.

Seeley, T.D., Visscher, P.K., Schlegel, T., Hogan, P.M., Franks, N.R., and Marshall, J.A.R. (2012). Stop Signals Provide Cross Inhibition in Collective Decision-Making by Honeybee Swarms. Science *335*, 108–111. DOI: 10.1126/science.1210361.

Shan, Q. and Mostaghim, S. (2020). Collective decision making in swarm robotics with distributed Bayesian hypothesis testing. In International Conference on Swarm Intelligence, Springer, pp. 55–67. DOI: 10.1007/978-3-030-60376-2_5.

Shan, Q. and Mostaghim, S. (2021). Discrete collective estimation in swarm robotics with distributed Bayesian belief sharing. Swarm Intelligence *15*, 377–402. DOI: 10.1007/s11721-021-00201-w.

Slobodkin, L.B. (1961). Growth and Regulation of Animal Populations. New York: Holt, Rinehart and Winston. ISBN: 0486639584.

Soorati, M.D., Krome, M., Mora-Mendoza, M., Ghofrani, J., and Hamann, H. (2019). Plasticity in Collective Decision-Making for Robots: Creating Global Reference Frames, Detecting Dynamic Environments, and Preventing Lock-ins. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4100–4105. DOI: 10.1109/IROS40897.2019.8967777.

Stark, H.-U., Tessone, C.J., and Schweitzer, F. (2008a). Decelerating Microdynamics Can Accelerate Macrodynamics in the Voter Model. Physical Review Letters *101*, 018701. DOI: 10.1103/PhysRevLett.101.018701.

Stark, H.-U., Tessone, C.J., and Schweitzer, F. (2008b). Slower is faster: fostering consensus formation by heterogeneous inertia. Advances in Complex Systems *11*, 551–563. DOI: 10.1142/S0219525908001805.

Talamali, M.S., Marshall, J.A.R., Bose, T., and Reina, A. (2019). Improving collective decision accuracy via time-varying cross-inhibition. In 2019 International Conference on Robotics and Automation (ICRA), pp. 9652–9659. DOI: 10.1109/ICRA.2019.8794284.

Talamali, M.S., Bose, T., Haire, M., Xu, X., Marshall, J.A.R., and Reina, A. (2020). Sophisticated collective foraging with minimalist agents: a swarm robotics test. Swarm Intelligence *14*, 1935–3820. DOI: 10.1007/s11721-019-00176-9.

Talamali, M.S., Saha, A., Marshall, J.A.R., and Reina, A. (2021). When less is more: Robot swarms adapt better to changes with constrained communication. Science Robotics *6*, eabf1416. DOI: 10.1126/scirobotics.abf1416.

Torney, C., Neufeld, Z., and Couzin, I.D. (2009). Context-dependent interaction leads to emergent search behavior in social aggregates. Proceedings of the National Academy of Sciences *106*, 22055–22060. DOI: 10.1073/pnas.0907929106.

Valentini, G., Ferrante, E., and Dorigo, M. (2017). The Best-of-n Problem in Robot Swarms: Formalization, State of the Art, and Novel Perspectives. Frontiers in Robotics and AI *4*. DOI: 10.3389/frobt.2017.00009.

Valentini, G., Hamann, H., and Dorigo, M. (2014). Self-Organized Collective Decision Making: The Weighted Voter Model. In Proceedings of the 13th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2014), pp. 45–52.

Valentini, G., Ferrante, E., Hamann, H., and Dorigo, M. (2016a). Collective decision with 100 Kilobots: speed versus accuracy in binary discrimination problems. Autonomous Agents and Multi-Agent Systems *30*, 553–580. DOI: 10.1007/s10458-015-9323-3.

Valentini, G., Brambilla, D., Hamann, H., and Dorigo, M. (2016b). Collective Perception of Environmental Features in a Robot Swarm. In Swarm Intelligence, Cham: Springer International Publishing, pp. 65–76. DOI: 10.1007/978-3-319-44427-7_6.

Yang, G.-Z., Bellingham, J., Dupont, P.E., Fischer, P., Floridi, L., Full, R., Jacobstein, N., Kumar, V., McNutt, M., Merrifield, R., et al. (2018). The grand challenges of Science Robotics. Science Robotics *3*, eaar7650. DOI: 10.1126/scirobotics.aar7650.

Yasa, I.C., Ceylan, H., Bozuyuk, U., Wild, A.-M., and Sitti, M. (2020). Elucidating the interaction dynamics between microswimmer body and immune system for medical microrobots. Science Robotics *5*, eaaz3867. DOI: 10.1126/scirobotics.aaz3867.