Path Formation and Goal Search in Swarm Robotics

by

Shervin Nouyan

Université Libre de Bruxelles, IRIDIA
Avenue Franklin Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
SNouyan@ulb.ac.be

Supervised by

Marco Dorigo, Ph.D.

Maître de Recherches du FNRS
Université Libre de Bruxelles, IRIDIA
Avenue Franklin Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
mdorigo@ulb.ac.be

August, 2004

A thesis submitted in partial fulfillment of the requirements of the *Université Libre de Bruxelles, Faculté de Sciences Appliquées* for the

DIPLOME D'ETUDES APPROFONDIES (DEA)

Abstract

In this work, we present a swarm robotic approach to exploration and navigation. Taking inspiration from swarm intelligence methods, we address the problem of solving complex tasks with the group of robots while using simple control strategies for an individual robot. In particular, our approach consists in visually connected robotic *chains*, where neighbouring members of a chain can perceive each other with a camera. A chain of robots can be used to establish a path between different locations, in this way allowing other robots to exploit the chain to navigate along the formed path. We present the results of two series of experiments. While in the first one we analyse the general capabilities of chain formation, in the second one the robots have to find a goal location and establish a path towards it starting from a home location. Three chain formation strategies are tested, differing in the degree of movement allowed to the robots which are aggregated into a chain.

Acknowledgments

First of all, I want to thank my supervisor Marco Dorigo for his advice and his patience. I want to thank Vito, Rodi, Tom, Halva, Elio, Christos, Max, Bruno, Mauro & Carlotta and all my other colleagues for making Iridia the nice and crazy place it is. I want to thank my family for giving me a warm place I can always return to. Last but not least, I want to thank Isabelle for her love and care.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this work we address the problem of controlling a robotic swarm to collectively solve exploration and navigation tasks. We want to apply swarm intelligence methods [6], which take inspiration from social insect colonies, to allow the swarm of robots to solve complex tasks, while using simple control strategies for an individual robot.

In real ant colonies the problem of exploration and navigation is solved by establishing paths. This is done in a very simple and distributed manner. Ants lay trails of pheromone, a chemical substance that attracts other ants. Deneubourg *et al.* [13] showed that the process of laying a pheromone trail is a good strategy for finding the shortest path between a nest and a food source, thereby establishing a path that others can follow.

Inspired by this methodology of path establishment by pheromone laying, our approach to exploration is to use a *chain* of robots, a concept previously introduced by Goss *et al.* [22], where the robots themselves act as trail markers, or beacons, in place of pheromone trails. We define a robotic chain to be a sequence of robots, where two neighbouring robots can sense each other and the distance between them never exceeds a certain maximum sensing range. In our case, the robots can visually sense each other by means of an omni-directional camera.

Robotic chains can be described by five characteristics. First, robots can form a chain by following simple rules relying on locally perceived information only. Second, in particular for open environments, a chain of robots has the advantage that it keeps a connection to a base station, thereby limiting the risk of robots to get lost. Third, a robotic chain can establish connections between different locations, in this way allowing other robots to exploit these connections in order to navigate along them. Fourth, the dis-

tance between such locations can be bigger than the perceptual range of one robot. Thus, the group of robots forming a chain can collectively overcome the limitations of a single robot. Finally, the approach of robotic chains is scalable to large groups of robots without the need of a more complex control strategy, a quality that is fundamental to swarm robotics.

Combining these characteristics, robotic chains can be clearly distinguished from other exploration strategies. For instance, planner based systems, which often rely on map-learning and path-planning strategies [31], may enable a robot to memorize important features of the perceived environment, thereby avoiding that the robot gets lost and possibly enabling it to navigate between distant locations. On the other hand, for a robot to create an internal map representation of its environment, complex control strategies are required that rely on idiothetic sensors,[1] which provide internal information about the robot's movements. As such idiothetic sensors involve an integration process, they are subject to *cumulative error*. Their quality accordingly decreases continually. Furthermore, the control complexity increases rapidly when applied to groups of robots. At the other end of the control spectrum, purely reactive approaches to exploration may enable the use of simple control strategies, and are often scalable to large groups of robots [5], but they are neither able to avoid the risk for a robot to get lost in open environments, nor do they provide a mechanism to navigate between distant locations.

In this work, we present the results obtained from ongoing work of the SWARM-BOTS project.[2] In the following section, we introduce the SWARM-BOTS project in more detail, giving a detailed description of the project's goals and putting them into relation with our work. Afterwards, we summarize the contents of this report in Section 1.2.

1.1 The Swarm-Bots Project

The goal of the SWARM-BOTS project is the development of a new robotic system, called a *swarm-bot* [12, 14, 34, 33]. A *swarm-bot* is defined as an artifact composed of a swarm of *s-bots*. An *s-bot* has simple acting, sensing and computational capabilities, and can therefore only solve a limited class of problems. In a swarm of *s-bots*, on the other hand, the collectivity is able to overcome the limitations of an individual, in this way solving problems

[1] Idiothetic sensors are often referred to as propriocective sensors.

[2] A project funded by the Future and Emerging Technologies Programme (IST-FET) of the European Community, under grant IST-2000-31010.

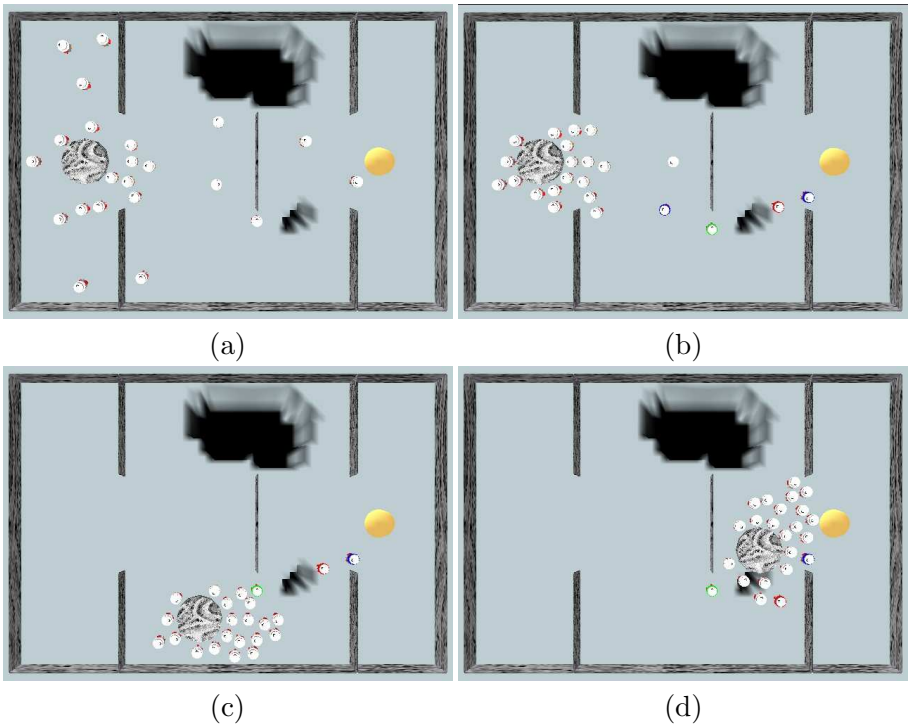


Figure 1.1: A scenario describing the goal of the SWARM-BOTS project.

that single *s-bot* cannot cope with. Swarm robotics can be considered as an instance of the more general field of collective robotics (see for instance [8, 26] for an overview of the field), which takes inspiration from the social insect metaphor and emphasizes aspects like decentralized control, simple control strategies for and individual robot, limited and local communication among the robots, and robustness.

The SWARM-BOTS project addresses the problem of developing control strategies that enable a swarm of robots to solve tasks such as coordinated motion, hole avoidance, collective transport of heavy objects, and—the subject of this work—collective exploration and navigation. A scenario (Figure 1.1) has been described that integrates these tasks and represents one of the main goals of the SWARM-BOTS project.

Figure 1.1(a) shows a swarm of *s-bots*, represented by white circles, which is situated in an environment containing a heavy object on the left part, and a goal location on the right part. Furthermore, the environment contains several hazards such as holes and walls. The *s-bots* have to transport the

object from its initial position to the goal location. The weight of the object is such that its transportation requires the coordinate effort of more than one *s-bot*.

As the goal location is not visible from the object, a path has to be established between the two locations. In Figure 1.1(b), this path is represented by a chain of four coloured *s-bots*. There are several possible paths connecting the initial and the goal location, which may have different lengths. The *s-bots* forming a chain have to choose the shortest connection and avoid the holes and the walls. The remaining *s-bots* aggregate around the object in order to collectively transport it and eventually reach the goal (Figures 1.1(c) and (d)).

Within this scenario, our work addresses the exploration of the environment in order to locate the object and the goal location, and the establishment of a path between them in the form of a robotic chain.

1.2 Report Layout

This report is organized as follows. In Chapter 2 we discuss the state-of-the-art in robotic exploration and navigation. In particular, we describe a hierarchy of navigation strategies based on the classification schemes of Trullier *et al.* [38] and Franz *et al.* [18]. Most of the strategies in the navigation hierarchy refer to the single robot domain. Therefore, we additionally give some examples that were applied to groups of robots, detailing previous approaches to chain formation and the differences between them and our work.

In Chapter 3, we give a description of the experimental framework we used. After introducing the hardware implementation and the simulation model of the *s-bot*, we describe the behaviour based controller that is used for the chain formation.

In a first set of experiments, which is the subject of Chapter 4, we try to reveal the general capabilities of a robot group performing chain formation by analysing the number of formed chains and their length when varying two probabilistic system parameters.

Then, we discuss the results of a second set of experiments in Chapter 5, where a group of 10 robots has to find a prey object that is placed in the environment, and establish a connection between the prey object and a base station.

Finally, in Chapter 6 we draw some conclusions from our experiments and indicate the future directions of our work.

Chapter 2

Robotic Exploration and Navigation

The term navigation originates from nautics and refers to the science and skill of sailing from one place to another. The navigator of a ship has to determine the ship's position, relate it to the desired destination, and accordingly set an adequate course for the ship. This description has entered into the domain of robotics nearly unchanged. For instance, Levitt *et al.* [24] define navigation by the following three questions: (i) "Where am I?"; (ii) "Where are other places relative to me?"; (iii) "How do I get to other places from here?".

The first question refers to the problem of localization, which is the process of identifying the robot's specific position. Answering this question does not necessarily have to yield the specific position within a global reference frame, but may more generally let the robot identify certain characteristics of its position. The second question denotes the process of putting the current position within a global representation of the environment. The answers to these two questions lays the basis for extracting the required actions to move towards a desired position, which is the object of the third question.

This interpretation of navigation is used by many robot navigation systems [23]. However, none of these systems has yet reached the flexibility and performance of animals such as bees, ants, birds or fish [18]. This has led robotics researchers to investigate more closely the navigation mechanisms applied in biological systems, which gave birth to the research field of *biomimetic robot navigation*. Navigation mechanisms in animals, the main source of inspiration for biomimetic navigation, do not necessarily rely on answering all or even any of the above mentioned three questions. On the

other hand, the most important issue appears to be the identification of how to reach the goal, which does not always require a localization or planning process.

In the following section, we introduce the hierarchy of biologically inspired navigation strategies as defined by Trullier *et al.* [38] and extended by Franz *et al.* [18]. Most of the discussed strategies refer to the single robot domain. Therefore, in Section 2.2 we discuss some implementations of exploration and navigation strategies in the multi-robot domain.

2.1 Navigation Hierarchy

Table 2.1 summarizes the six strategies in the hierarchy according to their behavioural prerequisites and navigation competences. The table is split into *local navigation* strategies and *way finding* strategies. Local navigation strategies have also been called tactics [41] or local control strategies [23]. An agent chooses its action on the basis of current sensory or internal information only, without representing any objects outside the current sensory horizon. Way finding strategies, on the other hand, also store and make use of global information.

2.1.1 Random Search

In the simplest form of navigation, a robot randomly explores the environment. A robot only requires the basic competences of locomotion and goal recognition. Compared to the strategies presented in the following sections, random search requires a large amount of time to detect the goal, but can be used as a backup strategy when the agent is not able to detect the goal.

2.1.2 Target approaching

Navigation would not be possible without the basic ability of approaching a perceived object. In biology, target approaching can be observed in most animals that are capable of locomotion. For a robot, to approach a target is a basic navigational requirement. To do so, the sensory information has to be used in order to orient the robot in the direction of the goal, often referred to as "body alignment". The robot must then be able to move toward the goal.

Braitenberg [7] shows that minimal sensory information and a very simple controller suffice to approach a target. Several studies address the target approaching behaviour in insects. For instance, Webb [27, 39] developed a

Table 2.1: The hierarchy of biologically inspired navigation strategies. Six strategies are classified with respect to the information they store and their characteristics.

Strategy	Used information	Characteristics
Local Navigation Strategies		
Random search	Goal recognition	Backup strategy
Target approaching	Goal recognition for body alignment	Basic requirement for navigation
Guidance	Extraction of goal direction from local landmark-configuration	Local navigation
Way Finding Strategies		
Recognition-triggered response	Set of landmark-configurations for each sub-goal	Global guidance
Topological navigation	Set of landmark-configurations linked by topological relationships	Topological detours
Metric navigation	Set of landmark-configurations linked by metrical relationships	Metrical detours Metric shortcuts

controller that mimics the sound approaching behaviour observed in female crickets, by using a mechanism that is able to discriminate the relative phase and the different travel times of incoming sound signals. Webb implemented this on a mobile robot that was able to find an artificial sound source [39]. This system was later extended so that the robot was able to find real crickets [27].

2.1.3 Guidance

Guidance is the process of extracting the direction towards a goal from the local landmark-configuration.[1] Bees and ants are able to use visual guidance to find a goal location which is only defined by an array of locally visible

[1]Landmarks, also referred to as beacons, are usually tall objects that can be perceived from comparably far distances, or even globally in the environment.

landmarks (for a review see [9]). Experiments suggest that to do this, an insect needs to memorize a snapshot of the spatial relationship between itself and the landmarks when it is located at the goal position. Later, when it is searching the goal position, it attempts to move so as to replicate this view.

This simple form of guidance has inspired several robot implementations as it enables a robot to find a goal that cannot be directly perceived, without requiring a complex representation of the environment. For instance, Franz *et al.* [19] applied a snapshot-based guidance method using a miniature robot with a conical mirror camera. Robust performance was shown in a number of experiments in a realistic low contrast environment. Möller *et al.* [32] successfully implemented a similar method using the Sahabot 2 on a flat plane in the Sahara desert with four black cylinders as landmarks.

2.1.4 Recognition Triggered Response

Guidance is a local navigation strategy as it requires only to process the locally available information. On the other hand, recognition triggered response, requiring the global localization of the robot, is a way finding method. Recognition triggered response is in many ways similar to guidance, as it relies on the perceived landmark configuration. It can be considered as an extension of the simple guidance strategy as, instead of just memorizing one landmark configuration, a set of landmark configurations is saved, each one connecting two locations by means of local navigation. In order to associate the appropriate local navigation method with the current landmark configuration, this method not only involves the recognition of the goal, but also of the starting location. The sequence of recognition triggered responses leads an agent to follow a route step by step, where the arrival at one sub-goal triggers the next step. In this way, a robot can navigate between locations that cannot be reached by local navigation methods alone.

Insects can associate movement decisions with visual landmarks. Ants, for instance, may learn to always pass a landmark on the right side [10]. This association persists even when the order of the landmarks or their relative positions to the nest are changed. Bees are able to learn routes, that is, a sequence of recognition triggered responses [11].

Recognition triggered response has been used for numerous robotic navigation systems. Gaussier and Zrehen [20], for instance, presented a robot that learned associations between compass directions and landmark configurations. The landmark configurations were extracted from panoramic images obtained from a rotating camera. A place was characterized by a

sequence of local landmark views and bearings connected by camera movements. The system could find its goal from any position inside an office room. Other implementations of recognition triggered response methods can be found in [35, 37].

2.1.5 Topological Navigation

The recognition triggered response method is only capable to lead an agent always through the same sequences of locations. There is no planning involved in the navigation process, possibly causing problems, for instance, if a part of the route is blocked by an obstacle. In that case the robot would have to perform a random search to find a known place again. This can be avoided if the spatial representation of the environment is goal-independent. To do so, a robot needs to be able to detect whether different routes pass through the same place, and in case they do, merge them by *route integration*. Integrated routes then become a topological global representation of the environment, which can be expressed as a graph with vertices representing locations, and edges representing the local navigation method to connect two vertices. Typically stored are the locations of objects, corridors, rooms and entrances to such rooms. By planning alternative routes, an agent using topological maps can dynamically adapt its route when encountering obstacles.

Biological systems seem to construct topological representations by integrating routes in a bottom-up manner [25]. This ability has been observed in many animals, ranging from honeybees [16] to humans [21]. Implementations on robots mostly follow such a bottom-up approach and mainly differ in the used place recognition, local navigation and route integration strategies.

Matarić [30], for instance, developed a behaviour-based controller for topological navigation. In contrast to most other approaches, the recognition of places in the environment was only determined by their context, that is, by the sequence of actions preceding the current one. The only information stored in the topological graph representation were actions, not place descriptions. The robot was capable of acquiring routes autonomously by following the walls of the experimental room. Routes were integrated as soon as the robot encountered previously visited locations. Mallot *et al.* [28] used a miniature robot to explore hexagonal mazes. Between junctions, the robot travelled by means of corridor following using infrared proximity sensors. Mallot *et al.* did not integrate views into a common place representation. Instead, the view graph was learned by a neural architecture that

associated sequences of views with movement decisions.

2.1.6 Metric Navigation

While for topological navigation a robot only memorizes key locations in the environment, metric navigation requires to learn all known places and their position in a global reference frame. In contrast to topological navigation, where the spatial relations are known between two directly connected locations only, in metric navigation the spatial relationship between any two locations can be extracted. An agent using metric navigation is able to find new paths through unknown terrain, as the integration of the current location into the reference frame allows it to deduce the spatial relations to previously visited locations. This includes, for example, shortcuts and detours around obstacles.

As for topological navigation, there is a vast literature concerning implementations of metric navigation on real robots. For a detailed review, we refer to [17, 31].

2.2 Multi Robot Exploration and Navigation

The navigation strategies detailed in the previous section have been successfully applied to the single robot domain. For multiple robots, however, there are additional challenges as well as opportunities for exploration and navigation of an environment, that may require extended or different strategies. In this section, we give some examples of implementations in the multi-robot exploration and navigation domain. In particular, we detail the previous approaches to chain formation and discuss the differences with our approach.

The concept of robotic chains was introduced by Goss *et al.* [22]. The robots act as trail markers or beacons that can be perceived by other robots. Robots are initially positioned around an initial beacon (the nest) and randomly explore its neighbourhood up to a certain distance d_{max}. The robots are prevented from exploring areas that are farther than this maximum distance from the nest. If a robot reaches the border of this area, it becomes a beacon itself and communicates this to the other robots by emitting a signal, thereby allowing them to explore its neighbourhood as well. This process leads to the formation of one or more chains of robots. In order to give a direction to the chain and enable in this way other robots to navigate to its end or back to the nest, the signal emitted by a robot in a chain contains a number i indicating how many robot-beacons are between robot i and the nest. Figure 2.1 shows an example for a group of 10 robots forming two

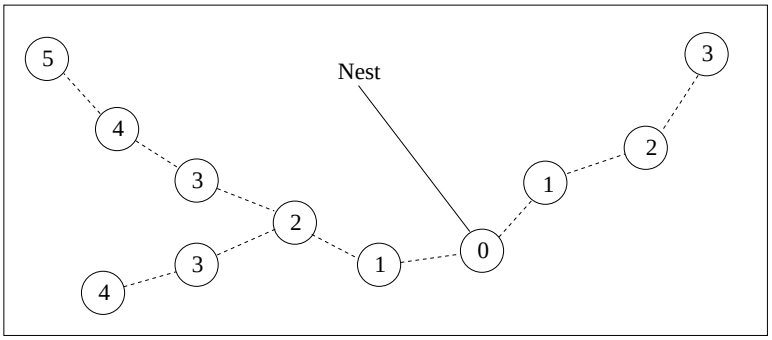


Figure 2.1: Robotic chains. Original concept of a directional robot chain by Goss *et al.* [22], where each robot has a number representing its global rank in the chain.

chains directly connected to the nest. Note that the left chain splits into two *branches*: branching of a chain occurs when more than one robot connect to a same robot-beacon.

Drogoul *et al.* [15] implemented in simulation several controllers inspired by the foraging behaviour of ants and extended these to obtain robotic chains that connect a nest with clusters of prey objects that have to be retrieved back towards the nest.

Werger *et al.* [40] used chains of real robots for a prey retrieval task as well. In their case, neighbouring robots within a chain sense each other by means of physical contact: one robot in the chain has to regularly touch the next one in order to communicate and maintain the chain.

Chapter 3

Chain Formation: Our Approach

The goal of this chapter is to give a detailed description of our approach to chain formation. It is therefore introductory to the following two chapters, where the experimental results will be presented. In the following section we give a definition of the chain formation problem. Then, in Section 3.2 we describe the differences of our work to the previous approaches to chain formation. In Section 3.3, we detail the experimental setup and introduce the hardware implementation and the simulation model of the *s-bot* we utilized in our experiments. The *s-bots* are controlled by a behaviour based controller. In order to form chains the robots have to perform different actions at different times. This is obtained by defining a set of states for a robot, each one activating a different set of behaviours. The *s-bot* controller is presented in Section 3.4, describing in detail the different states of our control system, and the respectively active sets of behaviours.

3.1 Problem Definition

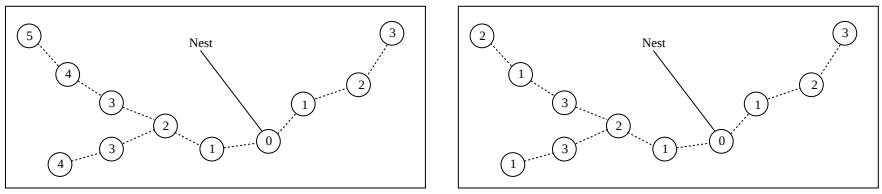
As mentioned previously, we are in general interested in developing control methods that enable groups of robots to collectively solve exploration and navigation tasks, and we have chosen chain formation as our basic methodology. As will be detailed in the following chapters, we use open environments without any borders. Initially, all robots are positioned in the proximity of a reference object which can be considered as the home or the nest of the robots. In order to explore the environment, the robots have to form one or more chains, each one consisting of a sequence of robots, where the distance

between two neighbouring robots never exceeds the maximum sensing range. A basic requirement for a robot chain is that neighbouring robots within a chain conserve their connection. This also involves a connection to the nest, which can be regarded as the root of each chain. Robots that are aggregated into a chain need to be distinguishable from other robots. Furthermore, a chain has to be directional in the sense that a robot that navigates along a chain is able to determine whether it moves away from or towards the nest. When a goal location is encountered by a chain, a connection has to be established, in this way forming a path that connects nest and goal, and that allows other robots to navigate between the two locations.

3.2 Differences to Previous Approaches

Adopting the idea of robotic chains from the previous approaches [22, 15, 40], we realized our system mainly modifying the original concept at four levels. The first important difference consists in the way the robots in a chain are numbered, as shown in Figure 3.1. In the original approach (Figure 3.1(a)) by Goss *et al.* [22] the chains are ordered with increasing numbers. On the other hand, in our approach (Figure 3.1b) the same shape of chains as in is ordered with a periodic sequence of three numbers. This can be done exploiting only local information—the state of neighbouring robots—and without the need of complex or symbolic communication, as will be shown in Section 3.4. The use of a sequence of three numbers to form a directional chain keeps the amount of information that has to be signalled by a robot in a chain constant. This makes it easy to signal the sequence of three numbers via, for instance, colours. In the original concept, on the contrary, the amount of information transmitted with such a signal, and thereby the complexity of the communication among the robots, increases for longer chains. Thus, we expect our concept to lead to a better scalability for larger group sizes.

The second difference of our work consists in the fact that Goss *et al.* used an extremely simplified, basically a point simulator without any modeling of embodiment, sensors or actuators. As opposed to this, we use a physics-based 3D simulator and a model of the *s-bot* that closely matches the attributes and behaviour of the real one, as tested for various settings [33]. Therefore, we believe that it will not be too difficult to validate our results on the real *s-bots* in the future. Werger *et al.* use real robots for their experiments. Nevertheless, their concept of chain formation relies on physical contacts between neighbouring robots by regularly touching each other. In



(a) (b)

Figure 3.1: Robotic chains. (a) Original concept of a directional robot chain by Goss *et al.* [22], where each robot has a number representing its global rank in the chain. (b) Our concept, in which each robot in the chain has one out of three numbers. The sequence of these numbers determines the direction of the chain.

order to be able to do this, neighbouring robots in a chain have to stay very close to each other, thereby significantly shortening the potential length of the chain. Additionally, a chain has to be aligned, eliminating in this way the possibility of branches in the chain. The possibility of branches in a chain is of fundamental importance for our work as we investigate basic attributes of the chain formation process such as the shape formed by a chain.

This leads us to the third difference of our work, which is reflected by our different goal. While Goss *et al.* [22] and Werger *et al.* [40] use the idea of chain formation for prey retrieval tasks, our ultimate goal is environment exploration. In particular, we aim at controlling the shape of the formed chains and the speed of the chain formation process by manipulating control parameters in an individual robot.

Finally, the last difference consists in the fact that we extend the original concept by introducing two additional control strategies that extend the capabilities of the formed chains. In the original system, members of a chain do not move at all. In a first robots that are aggregated into a chain extension, we allows minimal movement in order to adjust their positions such that the chain aligns itself. In a second extension, the members of a chain collectively move so that the formed chain as a whole explores the environment.

3.3 Experimental Setup

The *s-bot* has been designed within the SWARM-BOTS project. While an individual *s-bot* is very simple and limited in its actions, a swarm of *s-bots* should be able to efficiently overcome these limitations.

Figure 3.2: One of the two first *s-bot* prototypes.

As the real *s-bots* are still in the construction phase at the time of writing, we have conducted all our experiments in simulation. Two *s-bot* prototypes have been built and their specifications have been used to design the simulation software *Swarmbot3D*, based on the SDK Vortex^{TM} toolkit, which provides a 3D simulation that takes into account the dynamics and the collisions of rigid bodies.

3.3.1 Hardware Implementation

Figure 3.2 shows one of the first two hardware prototypes of the *s-bot*. The mobility of an *s-bot* is provided by a combination of two tracks and two wheels, which is called *Differential Treels*© *Drive*. It performs very well for straight motion, where tracks ensure a powerful displacement, and in sharp turns, where the wheels, bigger than the tracks and placed on a bigger radius, play a key role and ensure a very good rotation. Furthermore, treels lead to a good mobility on moderately rough terrain.

To enable an *s-bot* to grip an object or another robot, it is equipped with a rigid and a flexible gripper. The rigid gripper can also be used to lift objects, and is powerful enough to lift another *s-bot* if necessary. The flexible gripper is controlled by three servo-motors mounted on the *s-bot* turret, providing three degrees of freedom to extend the gripper, and to move it laterally or vertically.

For signalling purposes, each *s-bot* is provided with 24 LEDs—8 groups of red, green and blue LEDs—positioned on a ring around the robot. This LED ring is of particular importance for our work because, as mentioned

earlier, our concept of directional chains is based on signalling one out of three colours. Furthermore, each *s-bot* is equipped with a standard colour web-cam with a resolution of 640x480 pixels. A spherical mirror mounted on top of an *s-bot* allows a 360° panoramic view. As shown for instance by Marchese *et al.* [29], such a camera may be used to approximate the distance towards a perceived object with good accuracy. Using this camera, an *s-bot* can perceive the presence of other objects in the surrounding, particularly other *s-bots* signalling their state through their LED rings. In order to perceive its immediate vicinity, an *s-bot* is also endowed with 16 lateral infra-red proximity sensors.

The main processor is an ARM-based processor with a clock frequency of up to 400 MHz running a Linux operating system. Tests have shown that this processor can process simple algorithms on full colour images (640x480) in 100-200 ms. The power supply of an *s-bot* is ensured by two Lithium-Ion accumulators placed between the tracks. Given their capacity of 10Wh and an approximate power consumption of 3-5 W, the batteries should ensure continuous operation for at least two hours.

In addition to the mentioned features, an *s-bot* has various sensor and actuator devices such as for instance light sensors, ground sensors, directional microphones or a sound emitting system. For more information concerning the hardware implementation of an *s-bot*, we refer to the project web-site (`http://www.swarm-bots.org`) and to Mondada *et al.* [33].

3.3.2 Simulation Model

Given the hardware implementation, we have defined a simple *s-bot* model for running experiments in simulation. The simulation software *Swarm-bot3D*, based on the SDK Vortex^{TM} toolkit, provides the necessary functionalities to develop an accurate 3D dynamic simulation. The model, shown in Figure 3.3, reproduces all the important features of the prototype needed for our experiments. There is no need for physical connections between *s-bots* or between the *s-bots* and other objects, as for the formation of chains visual contact between the *s-bots* suffices. For this reason the grippers are omitted in our simulation model of the *s-bot*, thereby significantly increasing the simulation speed. The *s-bot* is modeled as a cylinder (radius: 6 cm, height 6 cm). The model is equipped with four spherical wheels (radius: 1.5 cm), two lateral and two passive wheels in the front and in the back. The lateral wheels are responsible for the motion of the *s-bot*. The two passive wheels model the balancing role of tracks, but, not being motorized, they do not contribute to the motion of the *s-bot*.

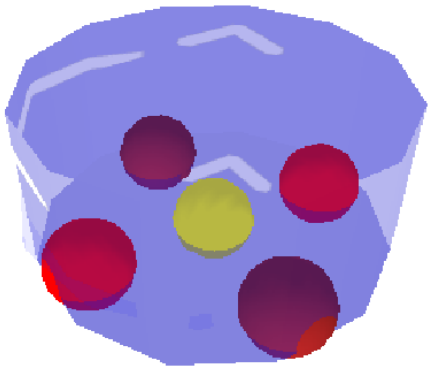


Figure 3.3: The *s-bot* model, reproducing all important features required for our experiments.

As will be shown later in this chapter, our control system relies on the detection of other robots that activated their LED ring in one specific colour. We expect to be able to extract this information from the environment with the omni-directional colour camera. In principle, it would be possible to simulate the camera by rendering a robots view.[1] However, simulating the camera in this way is computationally very expensive. Therefore, we use a simplified method to simulate the camera, in which we directly employ extracted features of the environment instead of processing the raw image. It is easy to access position and orientation of a simulated robot. These data can be used to compute the relative visibility between the robots, also taking into account the possibility for a robot to be shadowed and therefore not visible. In this way, the simulated camera collects the data and returns a vector with 360 entries, each one containing the approximate distance and colour of the first visible object for the degree. Some noise is added to both the perception of the distance and the colour.

For the purpose of collision avoidance, the control of an *s-bot* relies on the infra-red proximity sensors. They are simulated by utilizing a sampling technique based on data obtained for the Khepera robot [36], which has infra-red proximity sensors similar to those of the *s-bot*. Using a similar technique, we will soon collect samples from the *s-bot* proximity readings.

[1] In fact, in the SWARM-BOTS project there is ongoing work concerning this topic.

3.4 Controller

In this section, we describe the controller used by the robots to form chains. We start by qualitatively describing the different tasks involved in the formation of chains in the following section. Afterwards, an overview of the different states is given in Section 3.4.2. Finally, Section 3.4.3 details the behaviours executed by the robots in the different states and explains the specifications of the three different chain formation strategies we compared.

3.4.1 A Qualitative Description

In the introduction we described a scenario representing one of the goals of the SWARM-BOTS project. In the scenario, a group of *s-bots* has to find a prey item and establish a path to the nest that can then be exploited by other robots to navigate towards and retrieve the prey. This work addresses the search of the prey and the establishment of a path. As mentioned in the previous chapter, our approach consists in exploiting the local interactions between the robots, resulting in the dynamic formation of chains representing a path. In the following, we give a qualitative description of the behaviours governing the aggregation of the robots into chains.

Figure 3.4 presents a virtual scenario with ten robots at six different phases of chain formation. As shown in Figure 3.4(a), all robots are initially positioned around an object representing their home, the nest, and randomly explore its neighbourhood up to a certain distance. A prey item has to be found by the robots, and is positioned at a distance from the nest bigger than the perceptual range of one robot. The exploring robots, hereafter referred to as *explorers*, are prevented from exploring areas that are farther than a maximum distance from the nest.

Triggered by a probabilistic event, a robot may become a robot beacon, in this way allowing the other robots to explore its neighbourhood as well. A robot beacon, hereafter referred to as *chain-member*, communicates to the other robots by emitting a colour with its LED ring. This distributed process leads to the formation of one or more chains of robots, as indicated in Figure 3.4(b), where three robots activated their blue LEDs to signal that they are connected to the nest, which is in fact also perceived as a `chain-member`, but can be distinguished from other `chain-members` by its unique colour. The colour attracts `explorers` in the vicinity as they tend to move away from the nest.

In order to give a direction to the chain and enable in this way other robots to navigate towards its end or back to the nest, the colours emitted by

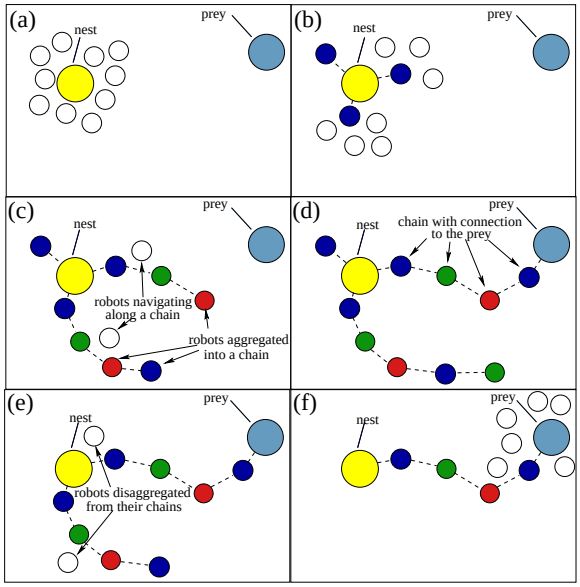


Figure 3.4: The images show six snapshots of a virtual scenario. The small circles represent the exploring robots (white) and the ones aggregated into a chain (blue, green or red). The formation of chains is shown from an initial situation (a) where all robots are positioned around the nest, represented by a larger yellow cylinder. In (b) and (c) several chains are formed until one of the formed chains eventually finds the prey in (d), thereby automatically establishing a path between nest and prey. In (e) the members of the chains that are not connected to the prey disassemble from the tail of a chain, and finally reach the prey in (f).

the robots in a chain are ordered in a periodic sequence of the three colours blue, green and red, as demonstrated in Figure 3.4(c). In Figure 3.4(d), one of the formed chains finds the prey and establishes a path between the prey and the nest, in this way enabling other robots to reach the prey by navigating along the chain.

Similar to the aggregation of an `explorer` to a `chain-member`, the disaggregation from a chain is also triggered by a probabilistic event. In Figure 3.4(e) two `chain-members` disaggregated from their chains and are moving back towards the nest.

The members of the chain which is connected to the prey, do not disaggregate as the last `chain-member` perceives the prey, thereby preventing the others from disaggregating. On the contrary, all other robots will release

themselves and reach the prey item by navigating along the only chain that is maintained. This is indicated in Figure 3.4(f). Note that in this way the robots reach the prey without being explicitly signalled that the prey was found or where it was found. No central controller is required, and communication among robots takes place only through signalling an internal state with the LED ring.

We designed our control system keeping the described desired behaviour in mind. In the next section we explain the state model that we employed.

3.4.2 The State Model

The robots have to perform different actions at different times. To realize this we defined a set of states for each robot. Depending on its state, a robot performs a different set of behaviours. As already mentioned in the previous section, we distinguish two main states: `explorer`, active when a robot navigates along a chain to explore the environment, and `chain-member`, active when a robot is aggregated into a chain. Furthermore, a third state called `lost`, is active when a robot has lost contact with a chain or with other robots.

The controller of a robot is discretized in time. The time step $t_{control}$ defines the length of the interval between the execution of two control sequences. The state of a robot is determined by its state during the last timestep and its current perception. Transitions between the states are triggered by probabilistic events and the local perception. Figure 3.5 gives an overview of the basic model. A circle represents a state and an arc in combination with a condition represents a switch from one state to another. The conditions are detailed in Table 3.1 and explained in the following.

In the next section we explain a basic model that contains the switches between the three main states. The basic model suffices to understand the basic rules governing the control system. However, the control of a robot is not purely reactive within a main state, but may differ depending on the past or current perception. In order to have a reactive control for a state, each main state is split into two sub-states. This leads to an extended state model with six states, which we refer to as the complete state model, and which is described in Section 3.4.2.2.

3.4.2.1 The Main States

At the beginning of an experiment, an *s-bot* is in the `explorer` state, and moves around the nest searching for chains that it can follow in order to get

away from the nest. If an *s-bot* does not perceive a chain it will with probability $P_{expl \rightarrow chain}$ per control time step $t_{control}$ connect itself to the nest and become a `chain-member` (*condition 1*). Otherwise, if it perceives a chain, the `explorer` will follow it until it perceives exactly one `chain-member`. This restriction is necessary as, for instance, two perceived `chain-members` could be aggregated into two different chains. Otherwise, when two perceived `chain-members` are aggregated into the same chain, a connection would create a loop between the three `chain-members` as our approach of chain formation is based on the sequence of three colours, each `chain-member` activating his LEDs in one of them. Thus, *condition 1* avoids inter-chain connections and loops within a chain. If one `chain-member` is perceived, a probabilistic event will trigger the `explorer` to aggregate into the chain.

Two conditions restrict the disaggregation of a robot from a chain. First, the `chain-member` should be situated on the tail of the chain in order to guarantee the stability of a chain, as otherwise parts of chains could loose contact to the nest. Second, the `chain-member` may not perceive any explorers in its neighbourhood. This ensures that `explorers`, which always rely on a `chain-member` to navigate and return back to the nest, do not loose contact to a chain (second part of *condition 2*). If these conditions are fulfilled, a `chain-member` will with probability $P_{chain \rightarrow expl}$ per control time step $t_{control}$ disassemble from the chain and become an `explorer`.[2]

In addition to this probabilistic disaggregation from a chain, we introduce another rule that can lead a `chain-member` to leave the chain and become an `explorer`, as expressed by *condition 3*. More details concerning this rule will be given in Section 3.4.3.2. Basically, *condition 3* expresses the possibility for two `chain-members` to merge if they have activated the same colours with their LED ring. This leads to an attraction of the two `chain-members` and causes one of them to disaggregate from its chain in case the distance between the two is smaller than a threshold d_{merge}.

Finally, a robot may lose contact with the group, thereby entering the state *lost*. We distinguish the case in which a robot cannot detect another robot or the nest (*condition 6*), or if it detects no `chain-member` (*condition 5*). If a robot resumes contact to a chain (*condition 4*), it enters the state explorer.

[2]We refer to the previous (next) chain colour as to the logically preceding (following) chain colour in the sequence of three colours shown in Figure 3.4

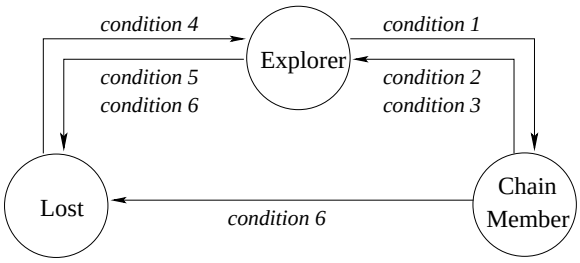


Figure 3.5: Transitions between the three main states. The states are represented by circles and the transitions by arcs. The conditions are detailed in Table 3.1.

Table 3.1: The six conditions concerning the transitions between the main states.

Condition	Explanation
condition 1	probabilistic event $P_{expl \rightarrow chain}$ and exactly one `chain-member` detected
condition 2	probabilistic event $P_{chain \rightarrow expl}$ and only previous chain colour detected
condition 3	distance between two `chain-members` of the same colour is smaller than d_{merge}
condition 4	`chain-member` detected
condition 5	no `chain-member`, but other robot detected
condition 6	no robot detected

3.4.2.2 Complete State Model

The previously introduced basic model describes the three main states and the switches between them. In the complete state model each of the main states is split into two sub-states, leading to a total of six states. In this section we explain the switches among the three pairs of states. The conditions for the transitions between the three main states remain as previously explained. Three conditions, summarized in Table 3.2, are added to express the internal transitions among the sub-states.

For a `chain-member`, the two substates distinguish whether the robot

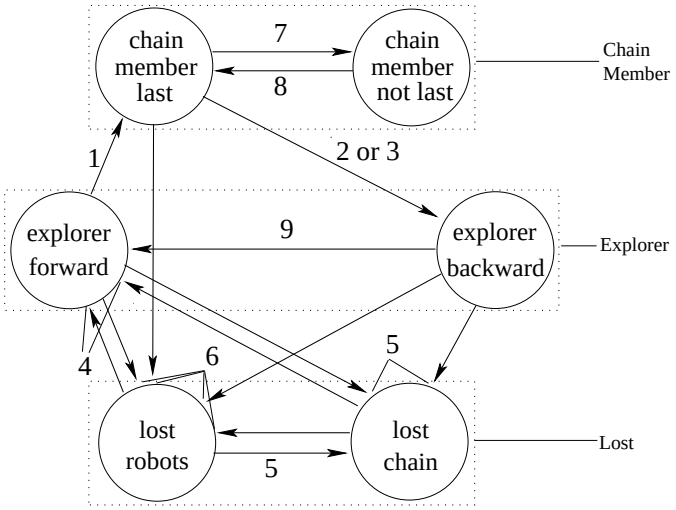


Figure 3.6: The complete state model. The six states are represented by circles and the transitions by arcs. Pairs of states belonging to the same main state are surrounded by a dotted box. The conditions are represented by numbers and detailed in Tables 3.1 and 3.2.

is the last member of its chain or not. This is decided on the basis of the perception of the next chain colour. If it is perceived (*condition 7*), the robot is in the state `chain-member` not last, otherwise (*condition 8*) it is in the state `chain-member` last.

For an `explorer`, the distinction is made for whether it moves along a chain in the direction away from the nest (*forward-explorers*) or back towards the nest (*backward-explorers*). Initially, an `explorer` moves away from the nest. After aggregating into and disaggregating from a chain, a robot re-enters the `explorer` state, and then moves back towards the nest. Once it perceives the nest (*condition 9*), it changes its internal state to find another chain that leads it away from the nest again. This mechanism aims at the dynamic creation of new chains and the destruction of old ones.

Finally, a `lost` robot is either in the state `lost-chain` or `lost-robots` based on whether it lost contact to all other robots (*condition 6*), or it lost contact to a chain but still perceives other `explorers` (*condition 5*).

3.4.3 Behaviours

All behaviours have been implemented following the motor schema paradigm [1, 2, 3, 4]. Within each state, a set of behaviours is active in

Table 3.2: The three conditions which trigger the transitions among the main states.

condition 7	next chain colour detected
condition 8	next chain colour not detected
condition 9	nest detected

parallel. Active behaviours are synchronized in time, computing independently for each control time step a vector that represents the desired direction of movement with respect to the robot's heading. The vectors of all active behaviours are added and the resulting vector is translated into movement of the wheels (see Section 3.4.3.4). In the following we detail the sets of behaviours that are active in each of the six sub-states. We employed three different strategies, *static*, *aligning* and *moving*, which only differ for the behaviours of a `chain-member`. Their specifications will be described along with the sets of behaviours for a `chain-member` in Section 3.4.3.2. All parameters introduced in this section are summarized in Table 3.3.

3.4.3.1 Explorer

The qualitative behaviours of the two `explorer` sub-states differ in either leading the `explorer` away from the nest (`forward-explorer`) or towards it (`backward-explorer`). Both states are implemented using the same three behaviours, differing only in the environmental context they refer to. Two of the behaviours, *adjust distance* and *move perpendicular*, are executed with respect to one `chain-member`. In order to select this `chain-member`, an `explorer` determines the two closest ones and then chooses one of them based on a lookup list as shown in Figure 3.7. A `forward-explorer` always chooses the `chain-member` that leads it away from the nest, while a `backward-explorer` respectively chooses the opposite one. If the two closest `chain-members` have identical colours, the robot chooses the closer one. This is the only difference between the two states.

After having chosen a `chain-member`, the behaviours *adjust distance* and *move perpendicular* each compute a vector, based on the relative position to the chosen `chain-member`. This is shown in figure 3.8, where the white and blue circles represent an `explorer` and a `chain-member`. Based on the relative heading α and the distance d of the chosen `chain-member`, *adjust*

Table 3.3: The most important parameters used by the controller.

Parameter	Explanation	Value
r_{s-bot}	*s-bot* radius	5.9 cm
d_{camera}	camera sensing range	50 cm
d_{chain}	desired distance between two neighbouring `chain-members`	40 cm
d_{expl}	desired distance between an `explorer` and his chosen `chain-member`	15 cm
v_{max}	maximum speed for an *s-bot*	$7.5\frac{cm}{s}$
$t_{control}$	control time step	100 ms
g_{ad}	gain value for *adjust distance*	5
g_{mp}	gain value for *move perpendicular*	1
g_{ac}	gain value for *avoid collisions*	1
g_{al}	gain value for *align*	1
g_{merge}	gain value for the *merge* behaviour	1
d_{lost}	desired distance for a `lost` robot to other detected robots	10 cm
d_{merge}	distance threshold for two `chain-members` to merge into one	3 cm

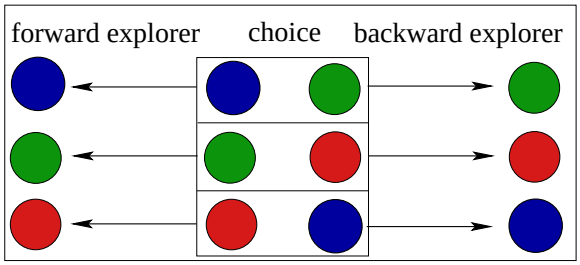


Figure 3.7: Mechanism for an `explorer` to choose one `chain-member`. The two coloured circles in the centre represent the colours of the two closest perceived `chain-members`. The arcs represent the choice for a `forward-explorer`, always choosing the one that leads it away from the nest, and a `backward-explorer`, choosing the opposite.

distance computes a vector F_{ad} that leads either to repulsion or attraction:

$$F_{ad} = \frac{d_{expl} - d}{d_{camera}} \cdot \begin{pmatrix} cos(\alpha) \\ sin(\alpha) \end{pmatrix}, \tag{3.1}$$

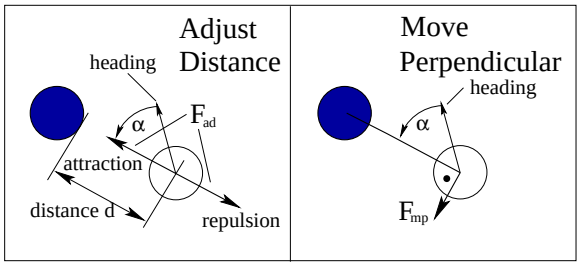


Figure 3.8: The two behaviours that are executed with respect to the relative position to the chosen `chain-member`, in this case a blue one. Based on relative heading α and distance d, *adjust distance* computes a vector F_{ad} that leads either to repulsion or attraction. The behaviour *move perpendicular* returns a vector F_{mp} that is right-angled in a clockwise sense to the `chain-member`. In combination, the two behaviours lead to movement that turns around the `chain-member` at a particular distance.

where d is the current distance from the `chain-member`, $d_{expl} = 15\ cm$ is the desired distance, and $d_{camera} = 50\ cm$ is the camera sensing range. If $d < d_{expl}$ the vector F_{ad} will point away from the `chain-member`, and vice versa.

The behaviour *move perpendicular* returns a vector F_{mp} that is right-angled in a clockwise sense to the blue `chain-member`, and therefore only depends on the relative heading α:

$$F_{mp} = \begin{pmatrix} -sin(\alpha) \\ cos(\alpha) \end{pmatrix}. \tag{3.2}$$

The third behaviour, *avoid collisions*, is controlled by the proximity sensors and returns a vector that leads to repulsion from objects that are too close. The normed activation A_j of a proximity sensor j results in a repulsion vector that is opposed to the source of activation:

$$F_{ac} = -\sum_{j=1}^{numProx} A_j \cdot \begin{pmatrix} cos(\beta_j) \\ sin(\beta_j) \end{pmatrix}, \tag{3.3}$$

where β_j denotes the direction of proximity sensor j with respect to the robot's heading.

The vector F_{expl} describes the overall behaviour of the `explorer` and is a weighted sum of the vectors given by the three behaviours:

$$F_{expl} = g_{ad} \cdot F_{ad} + g_{mp} \cdot F_{mp} + g_{ac} \cdot F_{ac}, \tag{3.4}$$

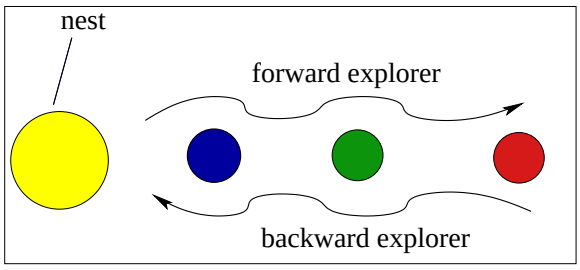


Figure 3.9: The difference between a `forward-explorer` and a `backward-explorer`. A chain of the robots, indicated by the small coloured circles, is connected to the nest. The arcs indicate the path taken by robots that are controlled by two different explorer states.

where the gain values are $g_{ad} = 5$, $g_{mp} = 1$, and $g_{ac} = 1$. The combination of the three behaviours leads to a movement that smoothly adjusts the robot's heading to turn around a `chain-member` at a certain distance, and at the same time avoids collisions with other objects in the vicinity. As indicated in Figure 3.9, these behaviours combined with the mechanism of choosing a `chain-member` cause an explorer to either move to the end of the chain or back to its root—the nest.

3.4.3.2 Chain Member

As mentioned earlier, we employ three different strategies for a `chain-member`, differing in the degree of movement involved in the behaviours.

In the simplest strategy, *static*, a `chain-member` may not move at all. The second strategy, *aligning*, allows minimal movement. Members in a chain align themselves so that the chain takes a linear structure. The most dynamic strategy, called *moving*, additionally allows movement of the last `chain-member`, so that a chain as a whole moves around the nest, thereby continuously exploring the environment. In the following the strategies are explained in more detail.

Static Strategy: The control for a static `chain-member` is very simple. When an `explorer` connects to a chain, it has to find the appropriate position, that is, a position at distance d_{chain} or greater, with respect to the previous member of the chain. Then, it activates the appropriate LEDs on its ring with respect to its position in the chain and thereby enters the state `chain-member`. The `chain-member` keeps its position and does not move

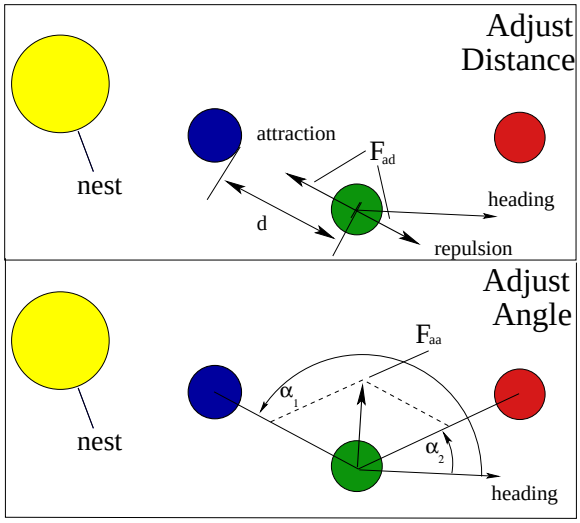


Figure 3.10: The two behaviours that are executed with respect to the relative position to two neighbouring `chain-members`. The *adjust distance* behaviour only differs from the one used for `explorers` by the distance constant. The *adjust angle* behaviour computes a vector F_{aa} that leads to a linear structure of the chain.

until it disaggregates from the chain. There is no difference between the behaviours of the two `chain-member` sub-states.

Aligning Strategy: The aligning strategy executes up to three behaviours depending on the robot's perception. Two of the three behaviours are executed with respect to the previous and the next member in the chain. These two behaviours are illustrated in Figure 3.10.

If the previous member is detected, the robot adjusts its distance to it with the *adjust distance* behaviour already used for `explorers`, differing only in the desired distance of $d_{chain} = 40cm$:

$$F_{ad} = \frac{d_{chain} - d}{d_{camera}} \cdot \begin{pmatrix} cos(\alpha_1) \\ sin(\alpha_1) \end{pmatrix}, \tag{3.5}$$

where α_1 is the angle to the previous `chain-member` with respect to the robot's heading.

If both a previous and a next member are detected, the *adjust angle* behaviour is executed in addition. It results in a movement between the two

neighbouring `chain-members`:

$$F_{aa} = \begin{pmatrix} cos(\alpha_1) + cos(\alpha_2) \\ sin(\alpha_1) + sin(\alpha_2) \end{pmatrix}, \tag{3.6}$$

where α_2 is the angle to the next `chain-member` with respect to the robot's heading.

A robot may perceive multiple robots that activated the previous or next chain colour. In this case it always chooses the closest ones. It is possible that no previous `chain-member` is detected. This may occur for instance if an `explorer` is shadowing the `chain-member`. In this case there is no reference point for any movement. Therefore, the robot remains still.

After some preliminary experiments we recognized that due to the movement implied in the behaviours for the aligning strategy, different chains may interact with each other in some way. We observed that an interaction between different chains may result in a chaotic behaviour of the `chain-members`, possibly lead to loops and the splitting of chains. We introduce a new rule (*condition 3*) that, as shortly mentioned before, may lead different chains to merge into one if their members perceive each other. The behaviour *merge* consists in attraction between two `chain-members` of the same colour, and the disassembly of one of the two `chain-members` in case the distance between is lower than a certain threshold distance d_{merge}. This is indicated in Figure 3.11, where two chains with three members merge into one:

(a) The members of the two chains perceive each other. Due to the *merge* behaviour each `chain-member` feels attracted to its counterpart in the other chain as expressed by the vector F_{me} with:

$$F_{me} = \frac{d_{camera} - d}{d_{camera}} \cdot \begin{pmatrix} cos(\alpha) \\ sin(\alpha) \end{pmatrix}, \tag{3.7}$$

where d is the current distance between two `chain-members` with the same colour and α is the relative heading.

(b) The distance between the two blue `chain-members` has reached the threshold d_{merge} so that the upper one of the two disassembles from its chain. In order to have an unambiguous rule for merging, the one that releases itself from the chain is always the `chain-member` that perceives its counterpart in a clockwise direction with respect to its previous neighbour, in this case the nest.

(c) The other two pairs of `chain-members` have merged as well as the distances between them are lower than d_{merge}. The robot that previously merged with the blue `chain-member` is already on its way back to the nest.

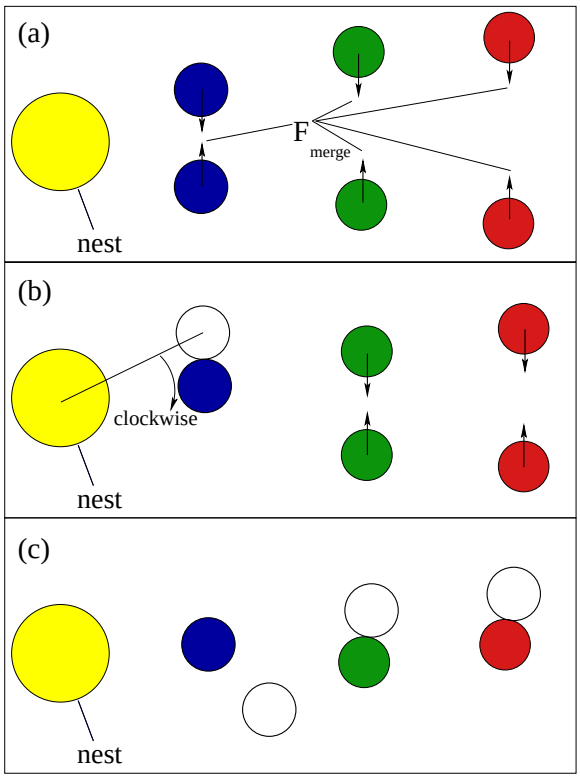


Figure 3.11: Three snapshots to illustrate the *merging* behaviour. (a) Two chains of each three robots are close to each other. Members of the chains signalling the same colour feel attracted to each other as indicated by the vectors F_{merge}. (b) The distance between the blue robots has undercut the threshold distance d_{merge}, so that one of the robots disassembles from the chain. In our rule, the one that releases itself from the chain is always the `chain-member` that perceives its counterpart in a clockwise direction with respect to its previous neighbour, in this case the nest. (c) The same process takes place for the other pairs of the chains too.

Finally, the already introduced *avoid collisions* behaviour is active for each `chain-member`. We obtain the vectors $F_{align,last}$ and $F_{align,notLast}$ that describe the overall behaviours for the aligning strategy:

$$F_{align,last} = g_{ad} \cdot F_{ad} + g_{ac} \cdot F_{ac} + g_{me} \cdot F_{me}, \quad (3.8)$$

$$F_{align,notLast} = g_{ad} \cdot F_{ad} + g_{ac} \cdot F_{ac} + g_{me} \cdot F_{me} + g_{al} \cdot F_{al}, \quad (3.9)$$

where the gain value g_{ad} is set to 5 and all other values are 1.

Moving Strategy: The moving strategy is an extension of the aligning strategy. One behaviour is added, and affects only the last member in a chain. While for the aligning strategy the last `chain-member` only adjusts its distance with respect to its precedent, in the moving strategy the *move perpendicular* behaviour also used for `explorers` is employed in addition. In this way the last `chain-member` turns around the previous one. As the rest of the chain continuously tries to align itself, the movement of the last member results in a clockwise movement of the whole chain around the nest. The last `chain-member` acts as a kind of leader that triggers the chain as a whole to move in a circle around the nest. The angular speed of a chain is determined by the speed of its last member and its length. However, the *move perpendicular* behaviour is only active for the last member of a chain when it perceives no `explorer`. As an `explorer` shadows the perception of a nearby `chain-member`, a `chain-member` can only be sure to be the last member of the chain when no `explorer` is perceived. This, however, has to be assured as the movement of a `chain-member` which is not situated at the tail of the chain can possibly break up the chain and leave a part of the chain without connection to the nest.

We obtain the vectors $F_{moving,last}$ and $F_{moving,notLast}$ for the two `chain-member` states:

$$F_{moving,last} = g_{ad} \cdot F_{ad} + g_{ac} \cdot F_{ac} + g_{me} \cdot F_{me} + g_{mp} \cdot F_{mp}, \quad (3.10)$$

$$F_{moving,notLast} = g_{ad} \cdot F_{ad} + g_{ac} \cdot F_{ac} + g_{me} \cdot F_{me} + g_{al} \cdot F_{al}, \quad (3.11)$$

where the gain values are set to $g_{ad} = 5$, $g_{mp} = 0.7$, and the value 1 for the others.

3.4.3.3 Lost

An `explorer` uses a chain to navigate in the environment, explores new areas and find a way back to the nest. Therefore, if it does not perceive any `chain-members`, it is `lost` as it has no reference point. A `chain-member`, on the other hand, is not necessarily lost in case it perceives no other `chain-members` as they might be shadowed by `explorers`. We discriminate a `lost` robot by whether it still perceives other `explorers` or not, and detail the respective behaviours in the following.

Lost Chain: If a robot perceives no `chain-member`, but one or more exploring robots, it tries to stay near them. The idea is that the perceived robots may have contact to a chain. Therefore, following them can lead the robot back to a chain. We implement this by executing the previously

discussed *adjust distance* behaviour for each detected robot, and summing them up:

$$F_{ad} = \sum_{j=1}^{numDetectedRobots} \frac{d_{lost} - d_j}{d_{camera}} \cdot \begin{pmatrix} cos(\alpha_j) \\ sin(\alpha_j) \end{pmatrix}, \tag{3.12}$$

where $d_{lost} = 10cm$ is the desired distance, d_j is the current distance, and α_j is the relative heading to a detected robot j. Additionally, the *avoid collisions* behaviour is active.

Lost Robots: In case no other robots can be detected, a robot does not move at all and waits until it detects other robots again.

3.4.3.4 Low Level Motor Control

Once the active behaviours have been summed up, the resulting vector F_{res} has to be translated into movement of the two wheels. This is done by the following function:

$$\begin{pmatrix} lSpeed \\ rSpeed \end{pmatrix} = \begin{cases} min\{|F_{res}|, 1\} \cdot \begin{pmatrix} cos(2 \cdot \alpha_{res}) \\ 1 \end{pmatrix} & 0 \leq \alpha_{res} < \frac{\pi}{2} \\ min\{|F_{res}|, 1\} \cdot \begin{pmatrix} cos(2 \cdot \alpha_{res} - \pi) \\ -1 \end{pmatrix} & \frac{\pi}{2} \leq \alpha_{res} < \pi \\ min\{|F_{res}|, 1\} \cdot \begin{pmatrix} -1 \\ -cos(2 \cdot \alpha_{res}) \end{pmatrix} & \pi \leq \alpha_{res} < \frac{3 \cdot \pi}{2} \\ min\{|F_{res}|, 1\} \cdot \begin{pmatrix} 1 \\ -cos(2 \cdot \alpha_{res} - \pi) \end{pmatrix} & \frac{3 \cdot \pi}{2} \leq \alpha_{res} < 2 \cdot \pi \end{cases},$$

where $lSpeed$ and $rSpeed$ denote the speed of left and right wheel, and α_{res} is the desired direction of movement with respect to the current heading.

Chapter 4

Chain Formation

In the last chapter we gave a detailed description of the control system. The goal of this chapter is to show the chain formation capabilities of the robotic swarm and to analyse the impact of two control parameters, namely the probability $P_{expl \rightarrow chain}$ for an `explorer` to aggregate into a chain and the probability $P_{chain \rightarrow expl}$ for a `chain-member` to disaggregate from one, on the group behaviour.

We present a first series of experiments that we conducted in order to test the control system, and evaluate it by observing the robot group while forming chains, in this way putting the focus on the analysis of the basic attributes of chain formation.

Section 4.1 explains the experimental setup, specifying the environment, the control parameters we applied, and the performance measures we used to evaluate the system. Afterwards, Section 4.2 discusses the results. Finally, Section 4.3 draws some conclusions and summarizes the results.

4.1 Experimental Setup

We employ the simplest possible environment, only consisting of the nest and the *s-bots* themselves. Walls are omitted as well as obstacles, holes or any other objects. By doing so, we can concentrate on analysing the features of chain formation without having to take care of any kind of environmental hazards. In the future, we will extend the capabilities of the control system to enable it to cope with more complex environments. However, also with the basic environment used for this work, we were able to extract interesting insights. Furthermore, for what concerns the basic analysis and the understanding of chain formation, we consider it to be advantageous to start with

such a simple environment.

In the following, we give a detailed description of the technical specifications we used for the first series of experiments.

4.1.1 Environment and Parameters

At the beginning of a trial, all *s-bots* are in the behavioural state `forward-explorer` and randomly positioned within a circle around the nest with radius $r_{init} = 50\ cm$. The nest is a cylindrical object with radius $r_{nest} = 12\ cm$, which is approximately twice the size of an *s-bot*. As the nest is initially within the camera sensing range d_{camera} of all robots, each *s-bot* can perceive it unless it is shadowed by another robot. Each trial is characterized by the two control parameters $P_{expl \to chain}$ and $P_{chain \to expl}$, and by a seed that initializes a random number generator to determine the initial positions of the robots, and their probabilistic choices during an experiment. For each of the two probabilities $P_{expl \to chain}$ and $P_{chain \to expl}$ ten different values are applied:

$$\begin{matrix} P_{expl \to chain} \\ P_{chain \to expl} \end{matrix} \in \{0.001; 0.002; 0.005; 0.01; 0.02; 0.05; 0.1; 0.2; 0.5; 1.0\},$$

resulting in 100 different combinations. For each combination, 100 seeds are used to initialize an experiment. An experiment runs for $t_{exp} = 1,000$ simulated seconds, corresponding to $10,000$ control time steps. Furthermore, we vary the number of robots, using a group size of either 5 or 10 robots. Table 4.1 summarizes the parameters.

Table 4.1: Complementing Table 3.3, the table shows the additional parameters concerning the experimental setup.

Parameter	Explanation	Value
r_{init}	radius from the nest concerning the circle within which the *s-bots* are initialized	$50\ cm$
r_{nest}	nest radius	$12\ cm$
t_{exp}	duration of one experiment	$1,000\ s$

4.1.2 Performance Measures

For this basic experimental setup, we analyse two attributes of the system: the number of formed chains and the largest distance from the nest covered by a chain. Both performance measures are recorded at the end of each experiment.

In view of using robotic chains for connecting different locations in the environment, the shape of the chains, mainly determined by the number of formed chains, is of fundamental interest. We aim at controlling the number of formed chains by varying the probabilistic control parameters. Different shapes may be advantageous for certain environmental conditions, and disadvantageous for others. For instance, if the *s-bots* form a single long chain, the advantage is that the chain can reach areas that are comparably far away. On the other hand, a single chain is directed towards one direction only, and therefore, unlike a system forming multiple chains, it does not thoroughly cover the area around the nest. The number of formed chains is computed by counting the number of `chain-members` directly connected to the nest.

The second performance measure is the distance between the nest and the farthest member of a chain. It is an important indicator for the efficiency of the system as it represents the potential length of a path that can be formed.

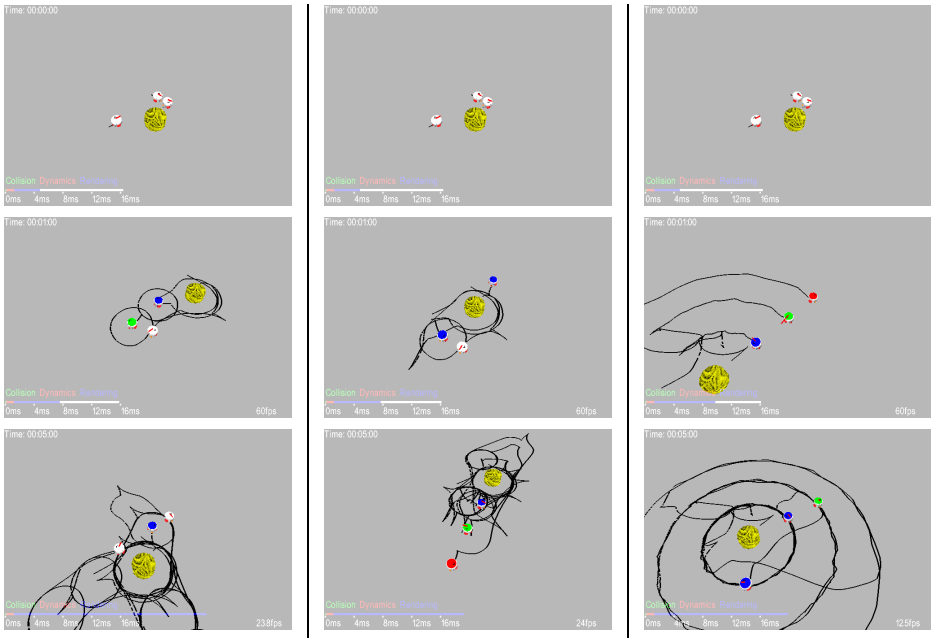
4.2 Results

In this section, we analyse the impact of the two probabilistic control parameters on the structure of the formed chains. Furthermore, we compare the three strategies, which differ in the amount of chain movement. In the simplest strategy, static, members of the chain cannot move. They act as immobile beacons that form a static path. As a first extension of this basic approach, the aligning strategy allows limited movement to `chain-members`. A `chain-member` can align itself with respect to the previous and next member of the chain, so that a linear chain is formed. In the last and most dynamic strategy, moving, the last member of a chain turns around its precedent `chain-member`. All other `chain-members` only align themselves. This leads to a coordinated collective movement of the whole chain around the nest.

4.2.1 Qualitative View on the Three Strategies

Before presenting the quantitative results of our experiments, we qualitatively describe the behaviour resulting from the three strategies. Figure 4.1 shows a selection of snapshots from a simulation with three robots applying (a) the static strategy, (b) the aligning strategy, and (c) the moving strategy. The snapshots are taken just after the *s-bots* were initialized (top row), after one minute (centre row), and after 5 minutes (bottom row). In order to give an impression about the explored area, the trajectories of the *s-bots* are displayed.

For each strategy, the robots are initialized at the same positions around the nest. After one minute, the robots have already formed one or two chains. While the chains of the static and the aligning strategy do not move, the chain in the moving strategy turns around the nest and has therefore



(a) (b) (c)

Figure 4.1: Snapshots of experiments with three robots controlled by (a) the static strategy, (b) the aligning strategy and (c) the moving strategy. The pictures are taken at the beginning of the experiments (top row), after one minute (centre row) and after five minutes (bottom row).

explored a larger part of the environment than the other two strategies. In the last row, as indicated by the trajectories, the *s-bots* controlled by the moving strategy have completed a whole circle while staying aggregated, in this way exploring all the area they could without getting disconnected from the nest. In the other two strategies, the exploration of the environment only takes place by destructing a chain and forming another one somewhere else.

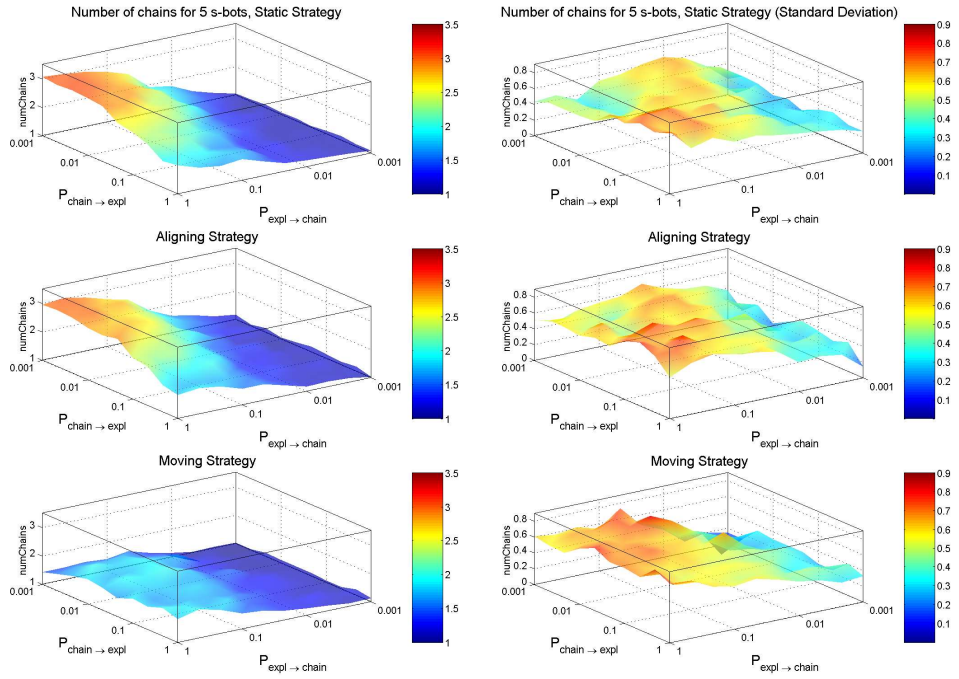
4.2.2 Number of Chains

Introducing our numerical results, Figure 4.2 displays the number of formed chains for the three strategies as a function of $P_{expl \to chain}$ and $P_{chain \to expl}$, and a group size of 5 *s-bots*. The two probabilities are scaled logarithmically on the horizontal axes. Both (a) the mean value and (b) the standard deviation over 100 repetitions are given.

Looking at the graphs, we can first of all recognize that for all control parameters and all strategies, the system forms in average between 1 and 3 chains. The results for the static and the aligning strategy are very similar. A maximum number of roughly three formed chains is reached for $P_{chain \to expl}$ close to 0, and $P_{expl \to chain}$ close to 1.

$P_{expl \to chain}$ determines the speed of the chain formation process. In other words, a high value for $P_{expl \to chain}$ leads to a system that forms chains very quickly. In fact, for $P_{expl \to chain} = 1$, a `forward-explorer` immediately aggregates into a chain in case only one `chain-member` is perceived. As there are no chains at the beginning of the experiment, the *s-bots* directly form as many chains as possible. The behaviour of a `forward-explorer` can then be described as *impatient* because it tends to become a `chain-member` as quickly as possible. On the other hand, when $P_{expl \to chain}$ is set to a low value, only one chain is formed in most of the cases. The `explorers` can be described as rather *patient*, as the low probability to aggregate into a chain causes the robots to search for existing chains rather than starting one. Note that an `explorer` requires approximately 10–15 seconds to make one complete circle around the nest. If the probability $P_{expl \to chain}$ is set to 10^{-3}, the expected time before it is triggered to aggregate into a chain is at least 50 s,[1] which is enough time to turn three times around the nest, and probably also enough time to find a chain, if there is any. Therefore, only one chain is formed in most cases for low values of $P_{expl \to chain}$.

[1] This results from $E(t_{expl \to chain}) \geq \frac{t_{control}}{2 \cdot P_{expl \to chain}}$. Only a lower bound can be given, because constraints related to the conditions that restrict a state transition may cause a further delay.



(a) (b)

Figure 4.2: Mean (a) and standard deviation (b) over 100 runs for the number of formed chains for a group of 5 *s-bots* and the three strategies. The two control parameters $P_{expl \rightarrow chain}$ and $P_{chain \rightarrow expl}$ are scaled logarithmically on the horizontal axes.

The second control parameter, $P_{chain \rightarrow expl}$, determines the speed of chain destruction, and therefore has a strong effect on the lifetime of a chain. For high values, `chain-members` disaggregate very quickly. In the extreme case of $P_{chain \rightarrow expl} = 1$ the last member of a chain immediately releases itself from the chain unless it perceives an `explorer`. In average, a lower number of chains is formed than for a system with a lower value of $P_{chain \rightarrow expl}$. In particular, shorter chains have a significantly shorter lifetime and therefore disband very fast in favour of longer ones. Therefore, fewer chains are formed. Nevertheless, $P_{chain \rightarrow expl}$ only has an impact on the number of formed chains when the other control parameter, $P_{expl \rightarrow chain}$, is set to a rather high value, where the group of robots in general tends to form more

chains.

For the moving strategy, the number of formed chains is in general lower than for the other two strategies. Usually, no more than two chains are formed, and for a wide area of parameter combinations there is just one. The main reason can be found in the merging mechanism, which causes an attraction between two `chain-members` of the same colour, so that one of the two may disaggregate from the chain. Even though the merging behaviour is also employed for the other two strategy, it has no impact there because the chains themselves are static, thereby preventing interactions between different chains. On the contrary, in the moving strategy chains encounter each other and merge frequently.

It is worth noting that for the moving strategy, low values of both probabilities lead to a very low standard deviation. This further confirms that the moving strategy nearly always leads to the creation of a single chain when using such experimental setting.

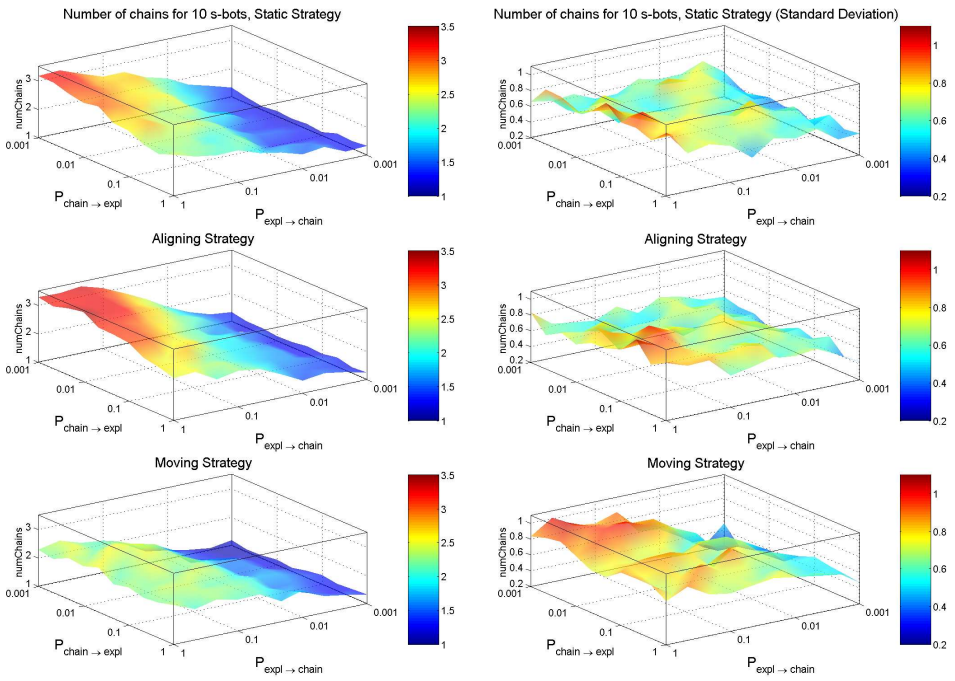
In Figure 4.3, the same measure is displayed for an increased group size of 10 *s-bots*. Similar to the results for 5 *s-bots*, the number of formed chains is roughly in the range $[1, 3]$ for the static and the aligning strategy, and in the range $[1, 2]$ for the moving strategy. As the number of formed chains does not scale with the robot group size, we can assume that that there is an upper bound to the number of formed chains. An `explorer` may only aggregate itself into a chain in case it perceives no more than one `chain-member` including the nest. For a certain number of chains around the nest, an `explorer` in the vicinity of the nest always perceives at least one `chain-member`. Therefore, if there are already several chains, an `explorer` cannot form a new chain and navigates along one of the existing chains. For the given nest size, our experiments have shown that in most cases three chains suffice to prevent the creation of new ones. When using a bigger nest, more than three chains can be formed.

4.2.3 Distance from Nest

Concerning the number of formed chains, the results are very similar for the static and the aligning strategy. For our second performance measure, the distance of the farthest `chain-member` from the nest, there are significant differences between the two. This is shown in Figure 4.4, where the length of the longest chain at the end of each trial is displayed for a group size of five robots. The distances range from 70 to 160 *cm* for the static, and from 90 to 210 *cm* for the aligning and moving strategy. In all three strategies, the chains reach longer distances for smaller values of both control parameters.

A distance of 210 *cm* corresponds to approximately four times the sensory range of a single robot.

For all 100 combination of the two probabilities $P_{expl \to chain}$ and $P_{chain \to expl}$, the aligning strategy and the moving strategy reach higher distances from the nest than the static strategy. For the moving strategy this can be explained by the fact that there are in general fewer and longer chains. Nevertheless, for certain parameter combinations the number of formed chains is similar, but the distances differ. This difference arises from the *adjust angle* behaviour, which is executed for the aligning and the moving strategy. It results in a linear structure of the chains, which thereby reach locations that are farther away from the nest. On the contrary, for the static strategy, the structure of the chains is not necessarily linear and can take various forms.



(a) (b)

Figure 4.3: Mean (a) and standard deviation (b) over 100 runs for the number of formed chains for a group of 10 *s-bots* and the three strategies.

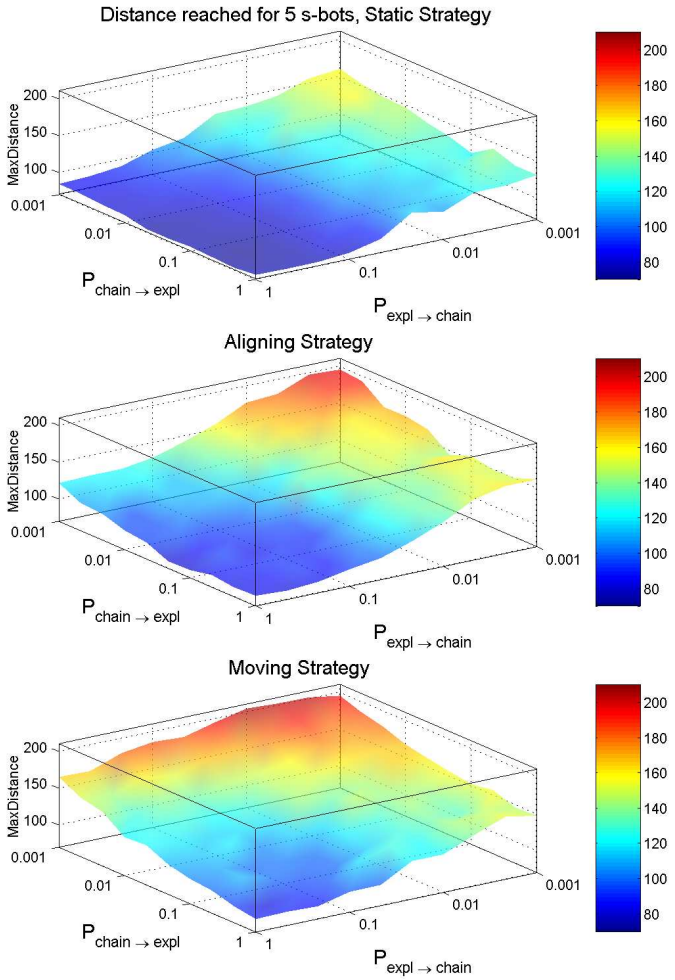


Figure 4.4: Longest distance covered by one chain for a group of 5 *s-bots* and the three strategies.

The aligning and the moving strategy reach approximately the same distances from the nest. However, for the moving strategy high distances are reached for a wider parameter range. In particular, for $P_{chain \rightarrow expl} = 10^{-3}$, the chains are always at least 160 cm long, while for the static strategy the length decreases to about 120 cm. This can be explained by considering the

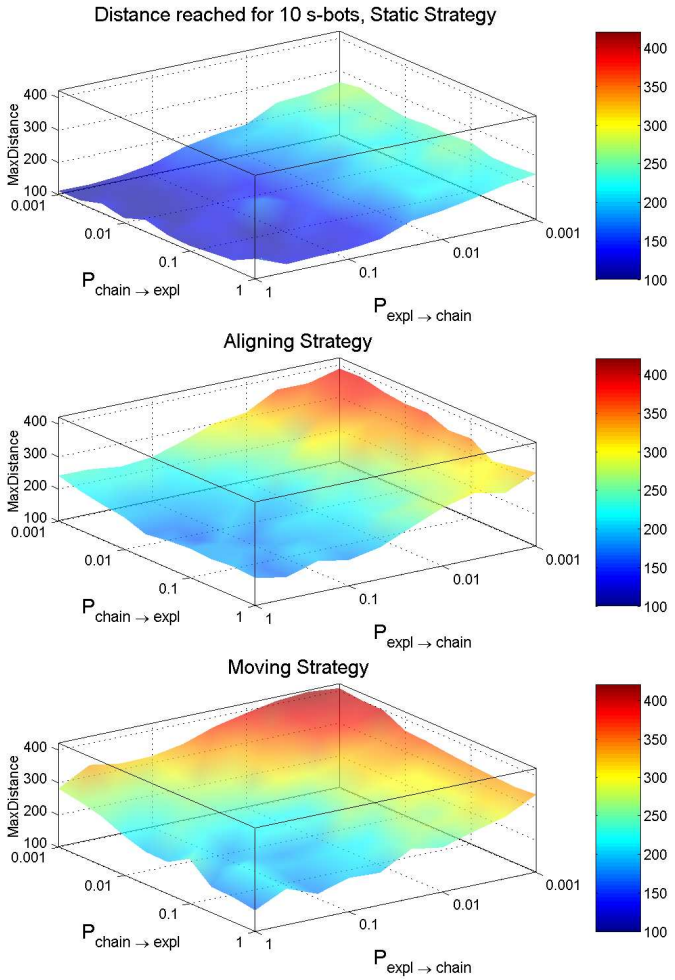


Figure 4.5: Mean over 100 values of the longest distance covered by one chain for a group of 10 *s-bots* and the three strategies.

merging mechanism, which results in a lower number of longer chains.

Figure 4.5 displays the distances reached by the three strategies for a group of 10 *s-bots*. For this increased group size, the distances range from 105 to 275 *cm* for the static, from 160 to 390 *cm* for the aligning, and from 160 to 420 *cm* for the moving strategy. The differences between the

three strategies are basically the same as for a group size of 5 robots, but unlike the number of chains, their length increases approximately linearly with a growing number of robots. This results from the more or less linearly increasing number of robots per chain.

4.3 Conclusions

In the presented results, we tried to reveal the general capabilities of a robot group performing chain formation. Governed by local behaviours, the individuals in the group aggregate into chains to collectively overcome the limitations of a single robot. The two control parameters have a significant effect on the overall behaviour of the robot swarm. In particular, low values for $P_{expl \rightarrow chain}$ result in a patient behaviour of the `explorers`, so that in most of the cases a single chain is formed. On the contrary, for $P_{expl \rightarrow chain}$ close to 1, the robots' behaviour can be considered as rather impatient, seeking to form many chains as fast as possible. The second control parameter, $P_{chain \rightarrow expl}$, determines the stability of the formed chains. Setting it high decreases the lifetime of the formed chains and increases the frequency of chain disbandment. By varying the two probabilities, several attributes of the global structure can be controlled. This concerns in particular the number and length of the formed chains, and the speeds of the processes that lead to the formation and the destruction of chains.

Among the three strategies, the moving strategy is the most active one, as the robots aggregated into a chain explore their environment by moving around the nest in circles. In the other two strategies, the chains are rather immobile. While for the static strategy a `chain-member` is restricted from any movement, `chain-members` controlled by the aligning strategy align themselves with respect to their neighbouring members in the chain, in this way forming linear structures. For these two strategies, the exploration of unvisited areas in the environment only takes place through the destruction of existing chains and the formation of new ones. Therefore, we expect the moving strategy to be the most effective one concerning the exploration of the environment. In order to test this, we conducted a second series of experiments, where we compare the performance of the three strategies when the group of robots has to find a goal item, as detailed in the following chapter.

Chapter 5

Goal Search

In this chapter, we present a second series of experiments, in which a group of ten *s-bots* has to find a goal item—the prey—placed at varying distances from the nest. We will describe the experimental setup in the following section. Afterwards, Section 5.2 will discuss the results of our experiments. Finally, we will summarize our insights and draw some conclusions in Section 5.3.

5.1 Experimental Setup

In this section we detail the specifications used for the second series of experiments and introduce the measures that we applied to assess the performance of the robot groups in each trial.

5.1.1 Environment and Parameters

Most of the parameters used for the previous experiments remain unchanged. The *s-bots* are initialized in the same circle around the nest, and in the state `forward-explorer`. We conducted our experiments for 100 combinations of the two control parameters $P_{expl \rightarrow chain}$ and $P_{chain \rightarrow expl}$, applying for each one ten values in the range $[0.001, 1]$. Again, each trial lasts for 1000 s. A cylindrical prey item is added to the environment, representing the robots' goal. Having approximately twice the size of an *s-bot*, the prey has the same radius as the nest. It is characterized by a unique colour, and can in this way be recognized by the robots. In all trials, the number of *s-bots* is fixed to ten. For this particular group size, we vary the difficulty of the experiment by positioning the prey at six different distances from the nest.

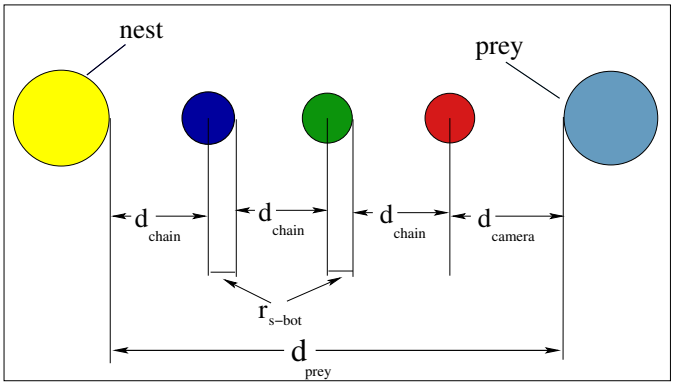


Figure 5.1: A chain of three *s-bots* forming a path between nest and prey. The distance d_{prey} is chosen in such a way that the chain has to be aligned and directed towards the prey in order to perceive it.

In the simplest setup at least three robots are required to aggregate into a chain to connect the nest with the prey. For each additional setups the prey distance was increased in such a way that one additional robot is required to aggregate into the chain that establishes the connection, leading to at least eight required robots for the largest prey distance.

Figure 5.1 illustrates the method we applied to choose the distance between nest and prey. The image shows a chain of three *s-bots* connecting nest and prey. The prey is located at a distance d_{prey} from the nest, such that the chain can only locate it in case it is directed towards the prey and all the `chain-members` are aligned. The distance d_{prey} depends on the distance between two neighboured `chain-members` d_{chain}, the *s-bot* radius r_{s-bot} and the camera sensing range d_{camera}, and can be calculated as follows:

$$d_{prey} = n \cdot d_{chain} + (n-1) \cdot r_{s-bot} + d_{camera}; \qquad n \epsilon \{3,4,5,6,7,8\},$$

where n is the minimum number of *s-bots* required in a chain to establish a path between nest and prey. For the given range of n, the distances vary between 180 and 410 cm, respectively requiring between 30 and 80% of the robots to be aggregated into the same chain pointing in the direction of the prey in order to find it.

5.1.2 Performance Measures

In the first series of experiments we discussed the basic capabilities of the three strategies, focussing on the number of formed chains and the length

they reach. The quantitative results revealed the differences between the strategies and the impact of the control parameters.

After having analysed these attributes of the chain formation process, the goal search task gives us the possibility to assess the potential of a robot group in exploring the environment by forming chains, and to compare this for the three strategies. We define two performance measures: success rate and completion time. The success rate represents the percentage of successful trials for a particular parameter combination and the 100 seeds. An experiment is considered to be successful if the robots are able to establish a path between nest and prey within 1000 s.

The completion time additionally considers the time required by the group of *s-bots* to locate the prey. For a successful run, the completion time is set to the time at which the prey was found. In case the robots do not find the prey, the completion time is set to 1000 s.

5.2 Results

Our results are sorted by strategy, first discussing the static strategy, then the aligning strategy, and finally the moving strategy. For each strategy, we present the average value of the success rate, and both the average value and the standard deviation of the completion time.

5.2.1 Static Strategy

For the static strategy we conducted experiments with three different experimental setups, placing the prey at either 180, 225 or 270 cm, respectively requiring at least three, four or five robots to be aggregated into a same chain in order to find the prey. The results are summarized in Figure 5.2, showing the average success rate, and in Figure 5.3, displaying the average completion time and its standard deviation.

When applying the easiest setup, 30% of the ten *s-bots* are required to connect nest and prey. For the best parameter combinations, the robots are able to successfully solve the task in approximately 95% of the cases, and in average within less than 250 s. However, the highest success rates diminish to 60% when the distance to the prey requires four robots in a chain to find it, and to 34%, when five robots are required.

The system performs best when the probability to disassemble from a chain, $P_{chain \rightarrow expl}$, is close to 1. This can be explained by the higher exploration speed related to higher values of $P_{chain \rightarrow expl}$. High values of $P_{chain \rightarrow expl}$ lead the robots to disassemble from the tail of a chain rather

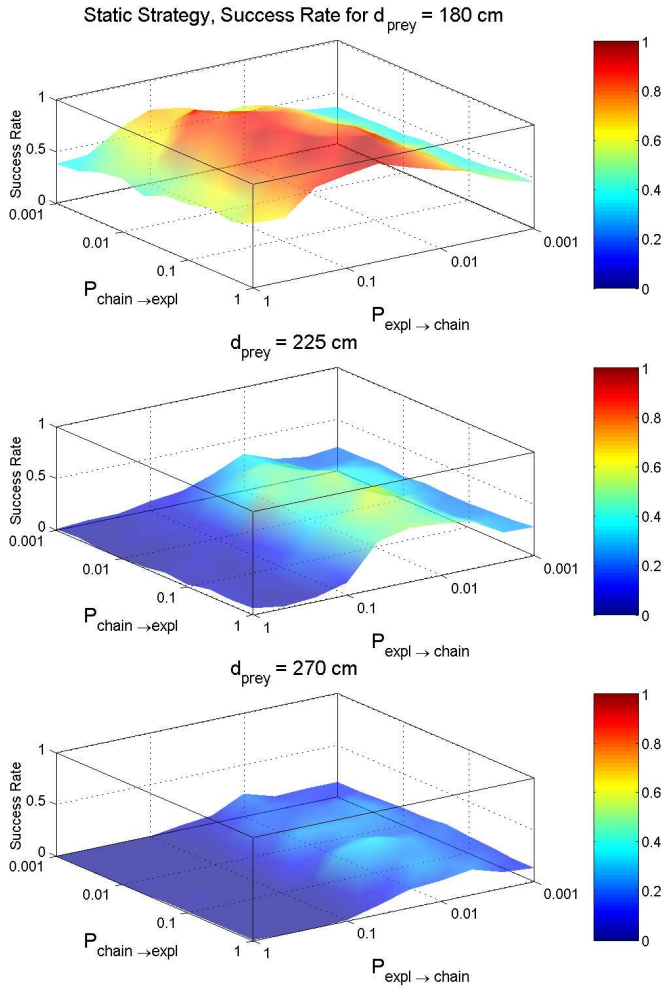
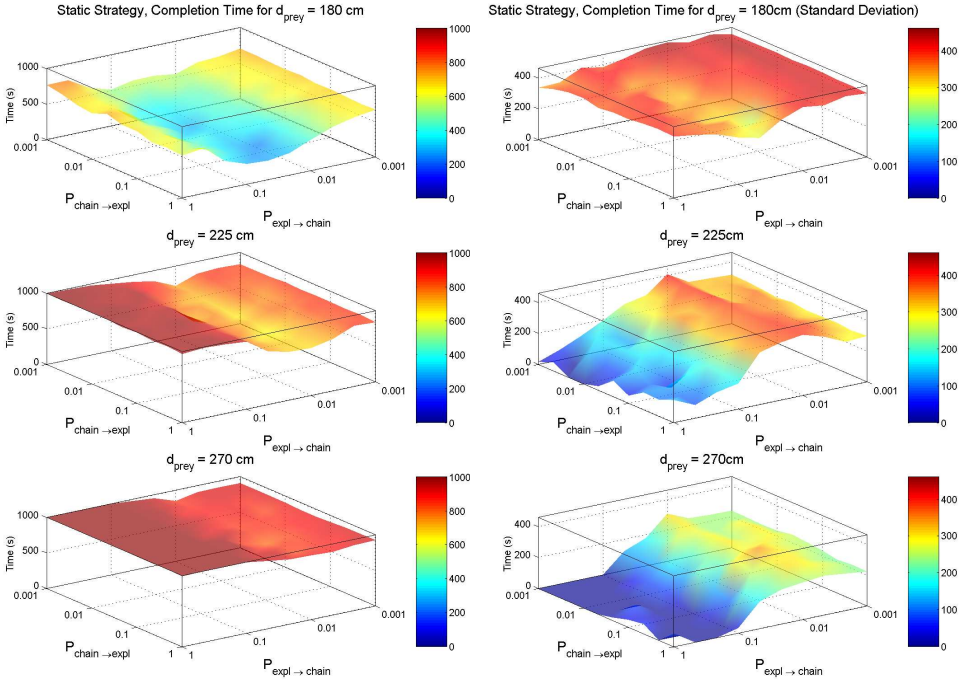


Figure 5.2: Success rate of finding the prey for the static strategy and three prey distances requiring at least three (d_{prey} = 180 cm), four (d_{prey} = 225 cm), or five (d_{prey} = 270 cm) *s-bots* to be aggregated into a same chain to find the prey.

fast. Consider for instance a chain that is not directed towards the prey. Such a chain is not able to locate the prey. Therefore, it is advantageous if such a chain is disassembled and a new chain is formed into a different



(a) (b)

Figure 5.3: Mean (a) and standard deviation (b) of the completion time to find the prey, the static strategy and three prey distances requiring at least three (d_{prey} = 180 cm), four (d_{prey} = 225 cm), or five (d_{prey} = 270 cm) *s-bots* to be aggregated into a same chain to find the prey.

direction. Low values of $P_{chain \rightarrow expl}$ increase the time it takes the system to give up chains and form new ones, thereby decreasing the efficiency in exploring unknown areas of the environment. Thus, high values of $P_{chain \rightarrow expl}$ increase the success rate of the *s-bots* to find the prey.

Interestingly, the most successful values of the other parameter, the probability to aggregate into a chain, $P_{expl \rightarrow chain}$, do not remain constant for different distances of the prey. While for the shortest distance between nest and prey, the values of $P_{expl \rightarrow chain}$ are around 0.1, they decrease to around 0.01 for increasing distances of the prey. This is not surprising when the results of the previous chapter are taken into account, where we have shown that the higher $P_{expl \rightarrow chain}$, the higher is the number of formed chains. When

the prey is placed far away from the nest, a high number of chains may be disadvantageous as, given the limited number of available *s-bots*, the chains could be too short to reach the prey. Therefore, for high prey distances, low values of $P_{expl \rightarrow chain}$ are more successful, leading to the formation of fewer and longer chains. For short prey distances, on the other hand, short chains may suffice to find the prey. Higher values of $P_{expl \rightarrow chain}$, leading to the simultaneous formation of several chains, are then more successful, as they lead the robots to explore different areas of the environment in parallel. Therefore, there is not one particular value of $P_{expl \rightarrow chain}$ that maximizes the performance of the system, but the value depends on the amount of robots that are required to aggregate into a same chain in order to establish a connection between nest and prey.

5.2.2 Aligning Strategy

For the aligning strategy we have conducted experiments using the same three setups as for the static strategy. The difference between the aligning and the static strategy consists in the behaviour of a `chain-member`. If it is controlled by the static strategy, a `chain-member` does not move at all. On the other hand, the aligning strategy leads a `chain-member` to adjust its position in order to reach a certain distance and angle with respect to its neighbours, resulting in the alignment of the chains.

Our results, which are summarized in Tables 5.4 and 5.5, show that this difference suffices to lead to a better performance of the robots in finding the prey. The aligning strategy outperforms the static one for all prey distances in both the success rate and the completion time. When the prey is 180 *cm* away, there is a wide range of parameter combinations that lead to high success rates. In particular, high values for both probabilities are very successful, leading to an average completion time of less than 200 *s*. When the prey is further away, the performance decreases, with maximum success rates of 78% for a prey distance of 225 *cm* and 56% for a prey distance of 270 *cm*. However, the decrease in performance is still more modest than for the static strategy, where the success rates drop to 60 and 34%.

As shown in the previous chapter, the alignment of the chains allows them to reach further distances from the nest when compared to the static strategy. Due to the alignment, a formed chain is linearly directed towards one direction. Therefore, two different chains with the same number of `chain-members` always reach approximately the same maximum distance from the nest. This is not the case for the static strategy, where a chain is not aligned, and therefore not linear. The distance reached by a chain then

depends on its shape. The alignment of the chains maximizes the distance reached from the nest, and in this way increases the efficiency in exploring the environment.

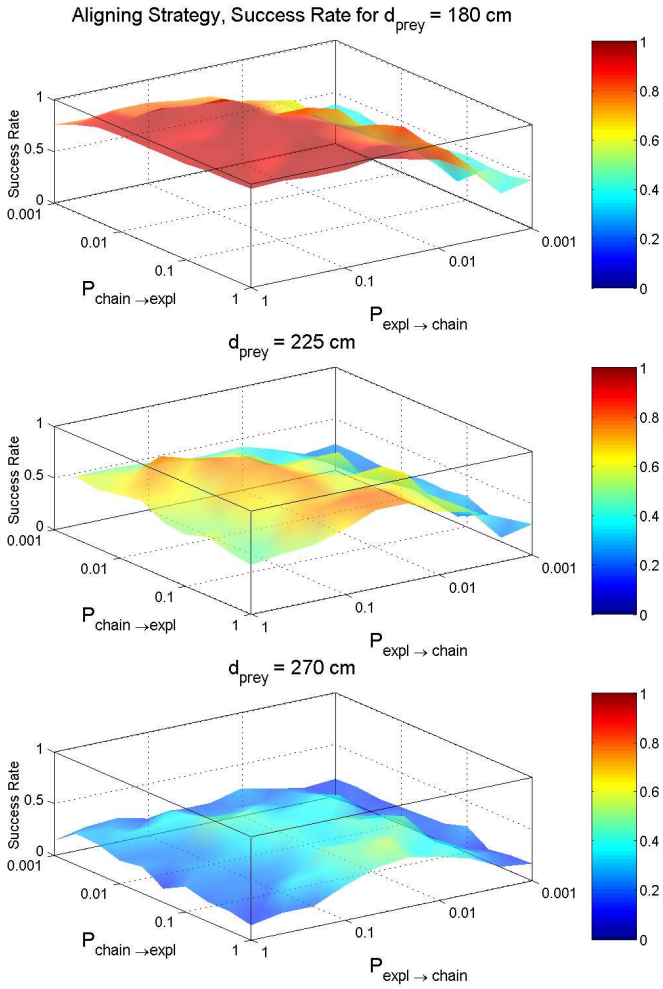
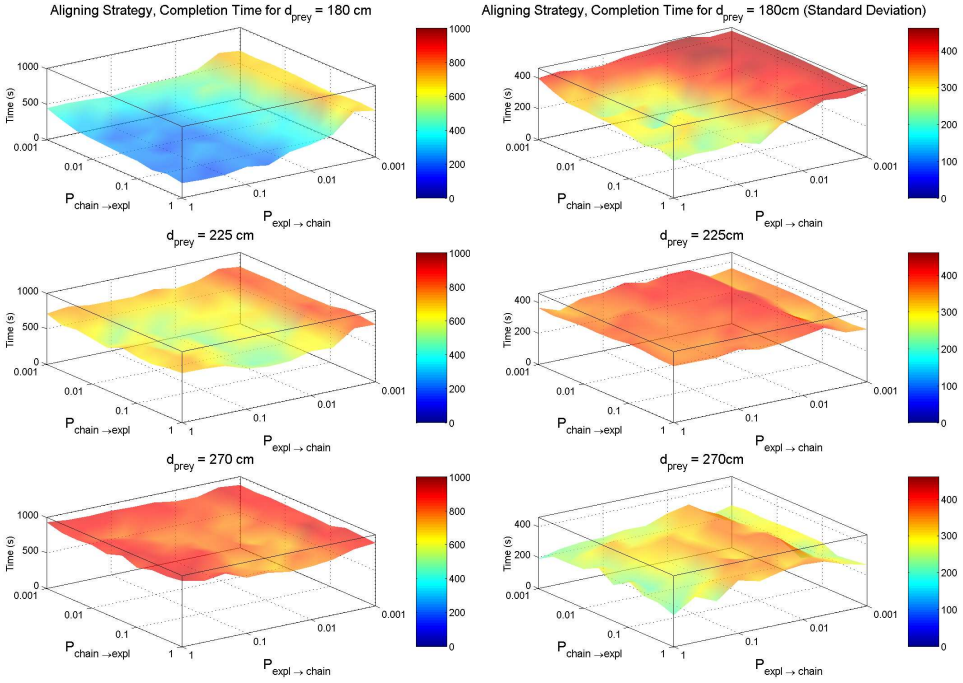


Figure 5.4: Success rate of finding the prey for the aligning strategy and three prey distances requiring at least three ($d_{prey} = 180\ cm$), four ($d_{prey} = 225\ cm$), or five ($d_{prey} = 270\ cm$) *s-bots* to be aggregated into a same chain to find the prey.



(a) (b)

Figure 5.5: Mean (a) and standard deviation (b) of the completion time to find the prey, the aligning strategy and three prey distances requiring at least three ($d_{prey} = 180\ cm$), four ($d_{prey} = 225\ cm$), or five ($d_{prey} = 270\ cm$) *s-bots* to be aggregated into a same chain to find the prey.

Despite the difference in performance, there are several similarities between the two strategies. For both, already formed chains do not contribute any more to a further exploration of the environment as the chains as a whole remain static. Exploration is then restricted to the destruction of chains and the formation of new ones into previously undiscovered directions. As this process is mainly governed by the two probabilistic parameters, the most successful combinations are very similar for the two strategies. The values of $P_{chain \rightarrow expl}$ are close to 1, which leads to a high destruction rate, and the most successful values of the second parameter, $P_{expl \rightarrow chain}$, decrease with increasing distances of the prey.

Low values of $P_{expl \rightarrow chain}$ decrease the number of formed chains, an

attribute that is advantageous if the robots have to find a prey that is far away. However, a low value of $P_{expl \rightarrow chain}$ also decreases the probability that a chain is formed into a new direction. Imagine, for instance, that all *s-bots* are aggregated into one chain. With probability $P_{chain \rightarrow expl}$ at each time step the last member of this chain will disaggregate and move back towards the nest. Once the robot perceives the nest, there are two possible actions: (i) it starts a new chain by connecting itself to the nest, or (ii) it moves around the nest until it perceives the chain from which it previously disconnected and then moves along this chain and finally reconnects to it. Which one of the two cases occurs depends solely on the time that the robot turns around the nest without being triggered to become a `chain-member`, which in turn is inversely proportional to the value of $P_{expl \rightarrow chain}$. The lower the value of $P_{expl \rightarrow chain}$, the lower is the probability that the robot starts a new chain, and the higher is the probability to move along the chain from which it previously disconnected.

This is particularly problematic for a task with a high prey distance, where a high percentage of the available robots have to be aggregated into the same chain in order to reach the prey. On the one hand, a low value of $P_{expl \rightarrow chain}$ is advantageous and even required, as it leads to the formation of fewer and longer chains. On the other hand, as stated above, a low value of $P_{expl \rightarrow chain}$ decreases the probability that new chains are formed into new directions. An already formed chain tends to persist rather than to disaggregate in favour of new ones, and in this way it blocks the exploration of the environment. Therefore, whether the prey is found or not strongly depends on the initial direction of the first formed chain.

5.2.3 Moving Strategy

The moving strategy differs from the other two strategies by allowing the formed chains to collectively move. This movement is led by the last member of a chain. While all behaviours of the aligning strategy remain active, the last `chain-member` executes one additional behaviour that leads it to turn around its predecessor in a clockwise sense. As all other `chain-members` adjust their position and angle with respect to their neighbours, the chain as a whole continuously realigns itself, in this way following the movement of the last member of the chain.

Being controlled by the moving strategy, the robots' exploration of the environment is not any more restricted to the destruction of already created chains and the recreation of new ones, as is the case for the static and the aligning strategy, but additionally takes place through the collec-

tive movement of a formed chain, which thereby actively contributes to the exploration process. Therefore, the moving strategy should lead to a higher efficiency in locating a prey item. This is confirmed by our results as shown in Figure 5.6, displaying the success rates, and in Figure 5.7, displaying the average and standard deviation of the completion time for the same experiments as conducted for the two other strategies with the prey distances of 180, 225 and 270 *cm*.

For the shortest prey distance there is a wide parameter range that leads to high success rates. Only when a small value for $P_{expl \rightarrow chain}$ of less than 0.01 is applied the performance drops below 70%. The best performing combinations of the two parameters lead the robots to find the prey, in average, in less than 100 *s*, compared to respectively 250 and 200 *s* for the static and the aligning strategy. Unlike the other two strategies, the moving strategy is still able to produce success rates of close to 1 when the prey distance is increased to 225 and 270 *cm*. The completion time then increases to approximately 200 and 320 *s*, compared to respectively 630 and 790 *s* for the static, and 480 and 680 *s* for the aligning strategy.

Given the better performance of the moving strategy, we conducted three additional experiments on more difficult setups where the prey distance was increased to 315, 360 and 405 *cm*, requiring at least six, seven or eight *s-bots* to be aggregated into the same chain to connect nest and prey. The success rate and completion time are summarized in Figures 5.8 and 5.9. The *s-bots* reach success rates of up to 93, 84 and 74 % in the three experiments. These values are obtained for $P_{chain \rightarrow expl} = 0.001$ and $0.005 \leq P_{expl \rightarrow chain} \leq 0.02$.

Opposed to the other two strategies, the *s-bots*' performance is maximized for a very low probability to disaggregate from a chain. As previously explained, for the other strategies exploration is restricted to the destruction and recreation of chains into new directions. Contrary to that, *s-bots* controlled by the moving strategy explore the environment mainly when they are aggregated into a chain by collectively moving around the nest. A high value of $P_{chain \rightarrow expl}$ is disadvantageous for two reasons. First, it disturbs the collective movement of the chain. Whenever a `chain-member` disaggregates from the tail of a chain, the chain as a whole stops to move. Remember that in order to maintain the stability of a chain, the last member of a chain may only move around its predecessor in case it perceives no `explorer`. If an *s-bot* disconnects from a chain, the new tail of the chain perceives this *s-bot* as `explorer` and may not move, thereby stopping the movement of the chain as a whole. Therefore, applying a low value for $P_{chain \rightarrow expl}$ maximizes the time that the chain moves and the efficiency in exploring the environment.

The second reason why a high value of $P_{chain \rightarrow expl}$ is disadvantageous

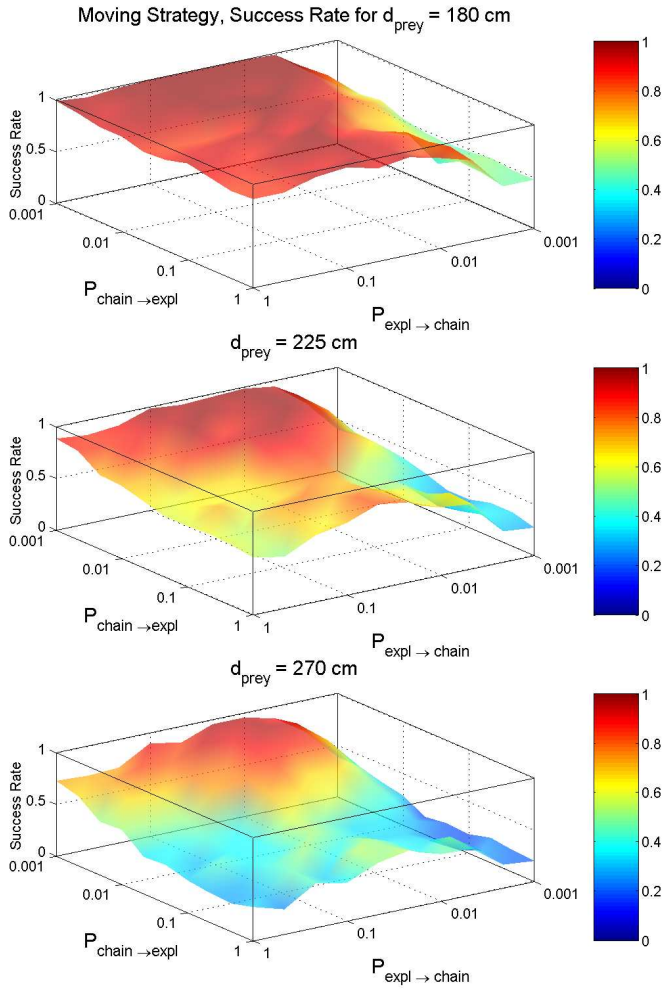
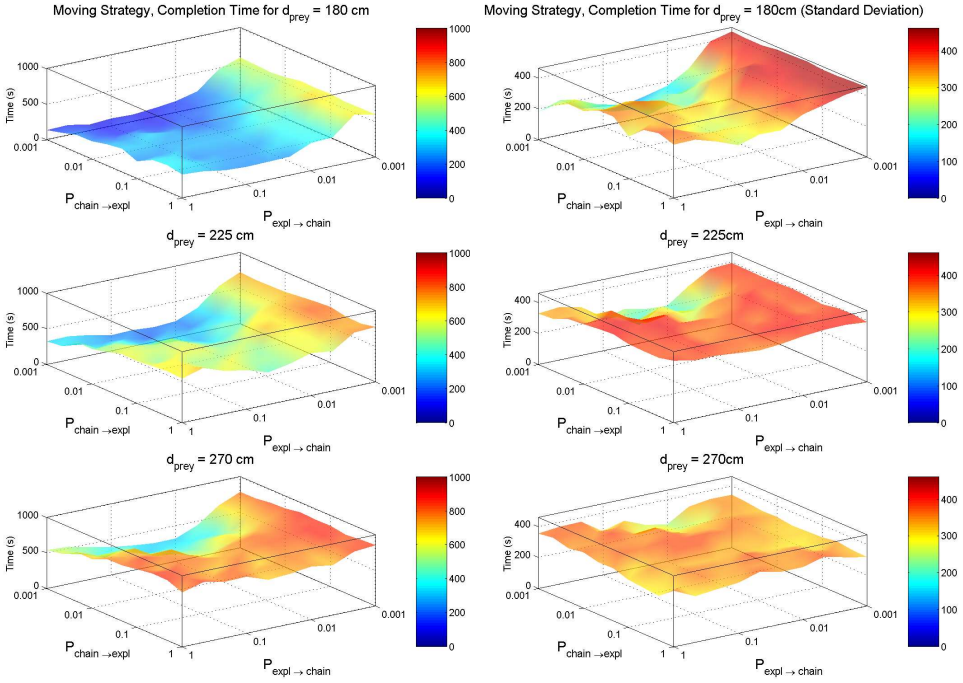


Figure 5.6: Success rate for finding the prey for the moving strategy and three prey distances requiring at least three ($d_{prey} = 180\ cm$), four ($d_{prey} = 225\ cm$), or five ($d_{prey} = 270\ cm$) *s-bots* to be aggregated into a same chain to find the prey.

is related to its impact on the length of a chain. As shown in the previous chapter, a higher probability to disaggregate from a chain leads to a decrease in the length of a chain. For short prey distances this may not be a problem



(a) (b)

Figure 5.7: Mean (a) and standard deviation (b) of the completion time to find the prey, the moving strategy and three prey distances requiring at least three ($d_{prey} = 180\ cm$), four ($d_{prey} = 225\ cm$), or five ($d_{prey} = 270\ cm$) *s-bots* to be aggregated into a same chain to find the prey.

because the prey can be found by a short chain as well. However, for high prey distances only values of $P_{chain \rightarrow expl}$ that maximize the length of a chain reach high success rates.

In the static and in the aligning strategy, the most successful values of the probability to aggregate into a chain decrease for increasing prey distances. This is partly true for the moving strategy, too. For short prey distances all values with $P_{expl \rightarrow chain} \geq 0.005$ reach a very high performance. However, the only values that lead to a high success for higher prey distances as well are 0.005 and 0.01. There are several reasons why these two values perform best. First, they lead the system to form one chain, which is an advantage as the interference between different moving chains merging into

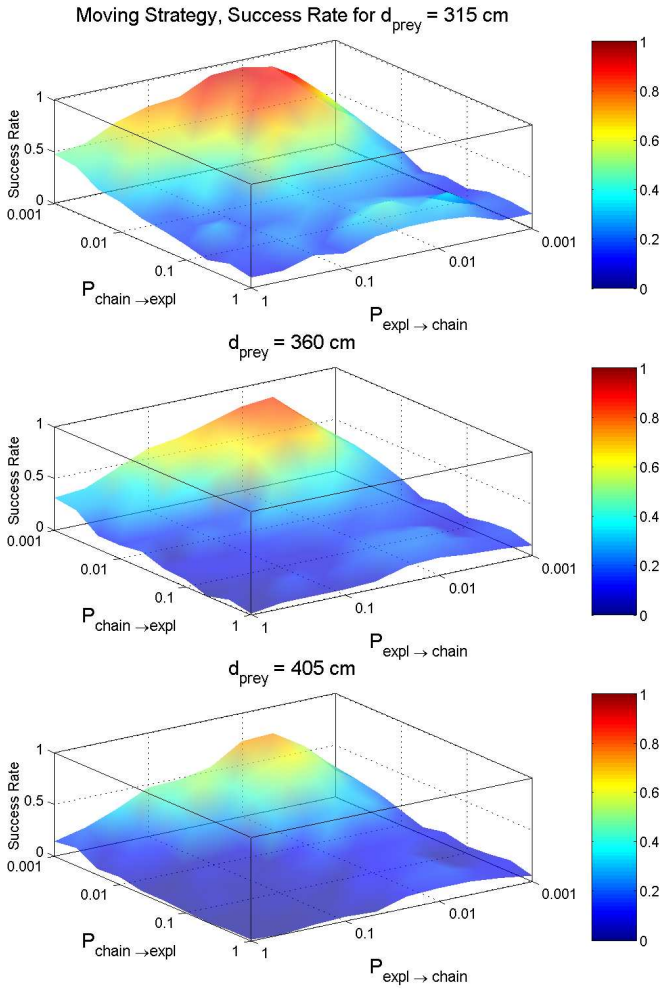
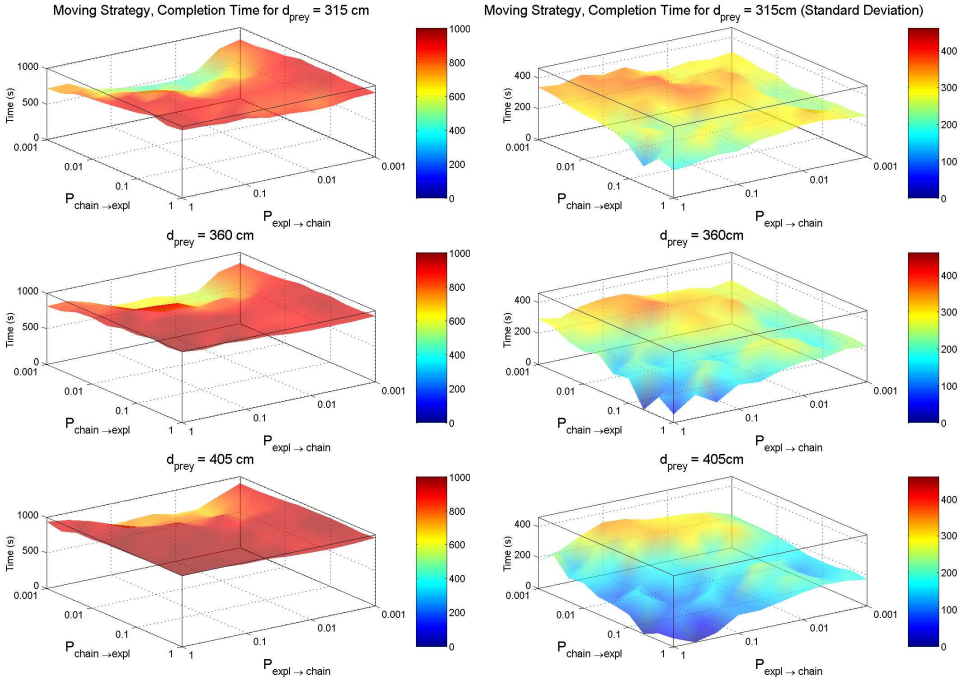


Figure 5.8: Success rate for finding the prey for the moving strategy and three prey distances requiring at least six ($d_{prey} = 315\ cm$), seven ($d_{prey} = 360\ cm$), or eight ($d_{prey} = 405\ cm$) *s-bots* to be aggregated into a same chain to find the prey.

each other results in a slowdown of the exploration. Another advantage is, again, the higher distance from the nest that can be reached by one single chain. These two arguments also hold for values of $P_{expl \rightarrow chain}$ lower than



(a) (b)

Figure 5.9: Mean (a) and standard deviation (b) of the completion time to find the prey, the moving strategy and three prey distances requiring at least six ($d_{prey} = 315\ cm$), seven ($d_{prey} = 360\ cm$), or eight ($d_{prey} = 405\ cm$) *s-bots* to be aggregated into a same chain to find the prey.

0.005, but for such low probabilities to aggregate into a chain, the process of chain formation significantly slows down. Thus, the two values 0.005 and 0.01 are the most successful ones because they result in the fastest formation of a single chain.

5.3 Conclusions

In this chapter we have presented experiments to reveal the capabilities of a group of 10 *s-bots* controlled by the three different chain formation strategies to locate a prey in the environment and establish a connection between the nest and the prey. We controlled the difficulty of the task by varying the

distance between the prey and the nest.

When the static or the aligning strategy are applied, already formed chains do not contribute to a further exploration of the environment because the chains as a whole do not move. Exploration is then restricted to the destruction of old chains and the creation of new ones into unexplored directions. This process is mainly governed by the two probabilistic parameters. In general, a high probability to disaggregate from a chain, $P_{chain \rightarrow expl}$, leads to a high destruction rate of formed chains and in this way to a higher success rate in finding the prey. The best performing values of the second parameter, the probability to aggregate into a chain $P_{expl \rightarrow chain}$, decrease for increasing distances of the prey. High values of $P_{expl \rightarrow chain}$ perform better for a short prey distance, as they result in the parallel formation of several chains which concurrently explore different directions from the nest. However, for higher prey distances, it is necessary that the robots aggregate into a single chain as otherwise the chain is too short to reach the nest. Therefore, lower values of $P_{expl \rightarrow chain}$ are more successful. Between the static and the aligning strategies, the latter performs better as the alignment of the chains results in farther distances that can be reached from the nest.

The moving strategy in general performs better than the other ones because robots aggregated into a chain contribute to the exploration process by collectively moving around the nest. In opposition to the other two strategies, a lower probability to disaggregate from a chain results in a higher success rate because low values of $P_{chain \rightarrow expl}$ decrease the frequency of situations in which a chain may not move, in this way increasing the time during which a chain can explore the environment. The most successful values of the probability to aggregate into a chain are in the range $[0.005, 0.01]$. Higher values lead to the formation of more than one chain, and lower values, while leading to a single chain, result in a slower chain formation process.

Chapter 6

Conclusions

In this work we have addressed the problem of collective exploration and navigation by a swarm of robots. Adopting the idea by Goss *et al.* [22], our approach consisted in the use of chains of visually connected robots. Chains of robots can be used to establish connections between different locations in the environment, and in this way enable other robots to exploit the connections in order to navigate between the locations. Based on this idea, we developped a behaviour based controller that makes use of local information only. Three different control strategies were implemented and analyzed. The static strategy, the most basic one, results in the formation of entirely static chains, where a robot does not perform any kind of movement once it is aggregated into a chain, which is a characteristic of all previous approaches to chain formation as well. The aligning strategy, a first extension to this basic approach, leads the members of a chain to adjust their position in order to reach a certain distance and angle with respect to their neighbours, resulting in the alignment of the chains. Finally, the moving strategy, further extending the aligning strategy, results in the collective movement of chains, which is led by the last member of each chain, that turns around its predecessor. As all other `chain-members` adjust their position and angle with respect to their neighbours, the chain as a whole continuously realigns itself, in this way collectively moving around the nest.

6.1 Experiments

In a first set of experiments we tried to analyse the general capabilities of a robot group performing chain formation. We varied two control parameters, the probability to aggregate into a chain $P_{expl \rightarrow chain}$, and the probability to

disaggregate from one $P_{chain \rightarrow expl}$. These two parameters have a significant effect on the overall behaviour of the robots. In particular, low values for $P_{expl \rightarrow chain}$ result in a patient behaviour of the exploring robots, so that in most of the cases a single chain is formed. On the contrary, for $P_{expl \rightarrow chain}$ close to 1, the robots' behaviour can be considered as rather impatient, seeking to form many chains as fast as possible. The second control parameter, $P_{chain \rightarrow expl}$, determines the stability of the formed chains. Setting it high decreases the lifetime of the formed chains and increases the frequency of chain disbandment. Furthermore, by varying the two control parameters, several attributes of the global structure can be controlled. This concerns in particular the number and length of the formed chains, and the speeds of the processes that lead to the formation and the destruction of chains.

Having analyzed the basic attributes of chain formation, we presented the results of a second set of experiments, in which the robots have to locate a prey object in the environment and establish a connection between the nest and the prey. We controlled the difficulty of the task by varying the distance between the prey and the nest. A comparison of the three control strategies has shown that for the given task the moving strategy in general performs better than the other ones because robots aggregated into a chain contribute to the exploration process by collectively moving around the nest. For the static and the aligning strategy, on the other hand, the exploration of the environment is restricted to the destruction of chains and the formation of new ones into previously undiscovered directions because already formed chains remain static.

6.2 Future Work

In order to understand the basic attributes of chain formation, we have so far analysed our system in rather simple environments. One of the first things we want to do in the future is to take into account more complex environments including obstacles such as walls, holes or objects to be avoided. Furthermore, we want to analyse the performance of our chain formation system in dynamic environments, where prey objects are dynamically placed in, and removed from, the environment.

In such dynamically changing environments, it would be interesting to use an adaptive algorithm. In particular, we are interested in finding a method which allows the robots to locally adjust the values of the two control parameters to a particular environmental situation so that for instance the robots learn to form longer and fewer chains in an environment where the

prey objects are placed far away from the nest, and shorter and more chains when the prey objects are close.

Another way for the robots to adapt to dynamically changing environments could be implemented by means of task allocation, where resources, that is, the robots, are dynamically allocated to find a prey only when this is required by the robot swarm, and otherwise remain in the nest. Therefore, we would want to analyse the impact of the number of robots on a particular task in order to find a way to measure the efficiency of the system and the optimal number of robots to be allocated.

Finally, an important issue that we want to address is to verify our simulation results on real robots as soon as these will be available.

Bibliography

[1] R.C. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behaviour. In C.K. Hemelrijk and E. Bonabeau, editors, *Proceedings of the IEEE Conference on Robotics and Automation*, pages 264–271, Raleigh, NC, 1987.

[2] R.C. Arkin. Motor schema-based mobile robot navigation. *International Jurnal of Robotics Research*, 8(4):92–112, 1989.

[3] R.C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[4] R.C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[5] M. Batalin and G. S. Sukhatme. Spreading out: A local approach to multi-robot coverage. In *Proceedings of the Sixth International Symposium on Distributed Autonomous Robotic Systems*, pages 373–382. Springer Verlag, Berlin, Germany, 2002.

[6] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NY, 1999.

[7] V. Braitenberg. *Vehicles*. MIT Press, Cambridge, MA, 1984.

[8] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:1–23, 1997.

[9] T. S. Collett. Landmark learning and guidance in insects. *Philosophical Transactions of the Royal Society of London*, B(337):295–303, 1992.

[10] T. S. Collett, E. Dillmann, A. Giger, and R. Wehner. Visual landmarks and route following in desert ants. *Journal of Computational Physiology*, A 158:835–851, 1986.

[11] T. S. Collett, S. N. Fry, and R. Wehner. Sequence learning by honeybees. *Journal of Computational Physiology*, A 172:693–706, 1993.

[12] E. Şahin, T. H. Labella, V. Trianni, J.-L. Deneubourg, P. Rasse, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, and M. Dorigo. SWARM-BOT: Pattern formation in a swarm of self-assembling mobile robots. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. IEEE Press, Piscataway, NJ, 2002.

[13] J.-L. Deneubourg, S. Aron, S.Goss, and J.-M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *J. Insect Behavior*, 3:159–168, 1990.

[14] M. Dorigo, V. Trianni, E. Şahin, T. H. Labella, R. Gross, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano, and L. M. Gambardella. Evolving self-organizing behaviors for a *Swarm-bot*. *Autonomous Robots*, 17(2–3):223–245, 2004.

[15] A. Drogoul and J. Ferber. From tom thumb to the dockers: Some experiments with foraging robots. In *From Animals to Animats 2. Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92)*, pages 451–459. MIT Press, Cambridge, MA, 1992.

[16] F. C. Dyer. Bees acquire route-based memories but not cognitive maps in a familiar landscape. *Animal Behaviour*, 41:239–246, 1991.

[17] D. Filliat and J.-A. Meyer. Map-based navigation in mobile robots - I. a review of localization strategies. *J. of Cognitive Systems Research*, 4:243–282, 2003.

[18] M. O. Franz and H. A. Mallot. Biomimetic robot navigation. *Robotics and Autonomous Systems*, 30:133–153, 2000.

[19] M. O. Franz, B. Schölkopf, H. A. Mallot, and H. H. Bülthoff. Where did i take that snapshot? Scene-based homing by image matching. *Byological Cybernetics*, 79:191–202, 1998.

[20] P. Gaussier and S. Zrehen. Perac: A neural architecture to control artificial animals. *Robotics and Autonomous Systems*, 16:291–320, 1995.

[21] S. Gillner and H. A. Mallot. Navigation and acquisition of spatial knowledge in a virtual maze. *Journal of Cognitive Neuroscience*, 10(4):445–463, 1998.

[22] S. Goss and J.-L. Deneubourg. Harvesting by a group of robots. In F. Varela and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life*, pages 195–204. MIT Press, Cambridge, MA, 1992.

[23] B. J. Kuipers and Y.-T. Byun. A robust, qualitative method for robot spatial learning. In *Proc. 7th National Conf. on Artificial Intelligence (AAAI-88)*, pages 774–779, Los Altos, CA, 1988. Morgan Kaufman.

[24] Tod Levitt and Daryl Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44:305–360, 1990.

[25] I. Lieblich and M. A. Arbib. Multiple representations of space underlying behavior. *Behavioral and Brain Sciences*, 5:627–659, 1982.

[26] J. Liu and J. Wu. *Multiagent Robotic Systems*, volume 21 of *International Series on Computational Intelligence*. CRC Press, Boca Raton, FL, 2001.

[27] H. H. Lund and B. Webb. A robot attracted to the cricket *Gryllus bimaculatus*. In P. Husbands and I. Harvey, editors, *Proceedings of the Fourth European Conference on Artificial Life*, pages 246–255, MIT Press, Cambridge, MA, 1997.

[28] H. A. Mallot, H. Bülthoff, P. Georg, B. Schölkopf, and K. Yasuhara. View-based cognitiv map learning by an autonomous robot. In F. Fogelman-Soulié and P. Gallinari, editors, *Proceedings of ICANN'95—International Conference on Artificial Neural Networks*, volume II, EC2, pages 381–386. Springer Verlag, Berlin, Germany, 1995.

[29] F. M. Marchese and D. G. Sorrenti. Omni-directional vision with a multi-part mirror. In *Proceedings of the Fourth International Workshop on RoboCup*, pages 289–298. Springer Verlag, Berlin, Germany, 2000.

[30] M. Matarić. Navigating with a rat brain: A neurobiologically-inspired model for robot spatial representation. In J. A. Meyer and S. W. Wilson, editors, *From Animals to Animats. Proceedings of the First International Conference on Simulation of Adaptive Behavior (SAB90)*, pages 169–175, MIT Press, Cambridge, MA, 1990.

[31] J.-A. Meyer and D. Filliat. Map-based navigation in mobile robots - II. a review of map-learning and path-planning strategies. *J. of Cognitive Systems Research*, 4:283–317, 2003.

[32] R. Möller, D. Lambrinos, R. Pfeifer, T. Labhart, and R. Wehner. Modeling and navigation with an autonomous agent. In R. Pfeifer, B. Blumberg, J. A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 5. Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB98)*, pages 185–194, MIT Press, Cambridge, MA, 1998.

[33] F. Mondada, G. C. Pettinaro, A. Guignard, I. V. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, L. M. Gambardella, and M. Dorigo. SWARM-BOT: A new distributed robotic concept. *Autonomous Robots*, 17(2–3):193–221, 2004.

[34] F. Mondada, G. C. Pettinaro, I. W. Kwee, A. Guignard, L. M. Gambardella, D. Floreano, S. Nolfi, J.-L. Deneubourg, and M. Dorigo. SWARM-BOT: A swarm of autonomous mobile robots with self-assembling capabilities. In C.K. Hemelrijk and E. Bonabeau, editors, *Proceedings of the International Workshop on Self-organisation and Evolution of Social Behaviour*, pages 307–312, Monte Verità, Ascona, Switzerland, September 8–13, 2002.

[35] R. C. Nelson. Visual homing using an associative memory. *Biological Cybernetics*, 65:281–291, 1991.

[36] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, 2000.

[37] M. Recce and K. D. Harris. Memory of places: A navigational model in support of Marr's theory of hippocampal function. *Hippocampus*, 6:735–748, 1996.

[38] O. Trullier, S. Wiener, A. Berthoz, and J. Meyer. Biologically-based artificial navigation systems: Review and prospects. *Progress in Neurobiology*, 51:483–544, 1997.

[39] B. Webb. Using robots to model animals: A cricket test. *Robotics and Autonomous Systems*, 16:117–134, 1995.

[40] B. Werger and M. Matarić. Robotic Food Chains: Externalization of State and Program for Minimal-Agent Foraging. In P. Maes, M. Matarić, J.-C. Meyer, J. Pollack, and S.-W. Wilson, editors, *From Animals to Animats, Proceedings of the 4th International Conference on Simulation of Adaptive Behavior*, pages 625–634. MIT Press, Cambridge, MA, 1996.

[41] S. Werner, B. Krieg-Brückner, H. A. Mallot, K. Schweizer, and C. Freksa. Spatial cognition: The role of landmark, route, and survey knowledge in human and robot navigation. In M. Jarke, K. Pasedach, and K. Pohl, editors, *Informatik 97*, pages 41–50. Springer Verlag, Berlin, Germany, 1997.