Heiko Hamann

Swarm Robotics: A Formal Approach

Springer

Swarm Robotics: A Formal Approach

Heiko Hamann

Swarm Robotics: A Formal Approach

Springer

Heiko Hamann
Institute of Computer Engineering
University of Lübeck
Lübeck, Germany

ISBN 978-3-319-74526-8 ISBN 978-3-319-74528-2 (eBook)
https://doi.org/10.1007/978-3-319-74528-2

Library of Congress Control Number: 2017964375

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

My major motivation to think about swarm robotics is the question of how probabilistic local actions of small robots sum up to rational global patterns shown by the swarm. For the engineer, it has potential to let a dream come true because complex problems may be solved by designing simple collaborating components. For the scientist, it has potential to help explain grand questions about consciousness, human societies, and the emergence of complexity.

Swarm robotics is a maturing field that deserves a book being dedicated to it completely. Until now there were many interesting books also treating swarm robotics besides other subjects, but there is no fully dedicated book yet. As a resource, especially for young researchers and students, a book seems important. I am trying to fill this gap. This book tells the story of designing maximally scalable and robust robot systems, which turns out to be challenging with today's methodology. Therefore, join me on our journey to find better and novel approaches to designing decentralized robot systems.

Like many other books, this one has a quite long history of origins. In 2013, I started to teach a quite special master's course for computer scientists called "swarm robotics" at the University of Paderborn, Germany. When I was creating the course, it was particularly hurtful not to have a book that teaches swarm robotics in a complete piece. Of course, there are books that contain material relevant to swarm robotics, such as the books on swarm intelligence by Bonabeau et al. [48] and Kennedy and Eberhart [210] and the great book on bioinspired AI by Floreano and Mattiussi [122]. However, they do not define a complete curriculum. So I had to go through what any teacher has to do when creating a course from scratch: define a canon of the most relevant subjects that each student needs to know and create all the teaching material. Probably back then I already had in mind that writing a book may help the field and be a reasonable option. This was certainly confirmed when students were desperately asking for accompanying reading material and I had to keep telling them: "Yes, that would be great but nobody has yet written that book on swarm robotics." So I started the long writing process slowly back in 2013 and students kept asking me for that book over the next years. In the years between

2013 and 2016, I taught that course four times and was able to improve the material. During the main writing period in 2017, I had moved on to another professorship in Lübeck, Germany; meanwhile, I was happy that I often could just write down what was already in my head from teaching the course. However, for a more complete picture of swarm robotics, it was necessary to add much more material going clearly beyond the content of a course. One can tell from the references in this book that I tried hard not to leave out relevant papers. It is still possible, however, or rather quite likely, that I have overlooked something or somebody maybe very relevant to swarm robotics. In that case, please accept my apologies and please let me know about it.

I hope that this book will help at least a few lecturers out there, in the case that they teach either a course fully dedicated to swarm robotics or a course that partially contains material on swarm robotics. You should not go through the same pains that I had when preparing a course out of nothing. Hopefully, also students find it useful, easy to follow, and maybe even a bit entertaining. Young researchers, who want to study the field and to do research on swarm robotics, will hopefully also find some useful information in here as a starting point. At least the problem of overlooking an important paper may be less likely. Finally, I also hope for enthusiastic swarm robotics hobbyists, who dare to look into such a rather scientific book with all that hard-to-follow terminology. Again, I have tried hard to make this text accessible pretty much to everybody or at least to those who know a bit about computer science or (computational) biology.

This book, of course, is only possible because there were many people who either helped me directly with this book or indirectly by discussing swarm robotics with me over the last decade or by letting me know about relevant new or old papers. I want to thank Marco Dorigo, Thomas Schmickl, Payam Zahadat, Gabriele Valentini, Yara Khaluf, Karl Crailsheim, Ronald Thenius, Ralf Mayet, Jürgen Stradner, Sebastian von Mammen, Michael Allwright, Mostafa Wahby, Mohammad Divband Soorati, Tanja Kaiser, Eliseo Ferrante, Nicolas Bredeche, Sanaz Mostaghim, Jon Timmis, and Kasper Støy. In addition, I want to thank my students attending the swarm robotics course in Paderborn between 2013 and 2016 for asking so many interesting questions, for sharing their solutions to problems of swarm robotics, for motivating me with their enthusiasm, and for challenging me with skeptical questions.

My sincere thanks go to my contacts at Springer Science+Business Media for their patience and for pushing me in the right moment: Mary E. James, Rebecca R. Hytowitz, Murugesan Tamilsevan, and Brian Halm.

Many thanks also for granting permissions to print photos: Thomas Schmickl, Francesco Mondada, Farshad Arvin, José Halloy, Ralf Mayet, Katie Swanson, and Martin Ladstätter.

Lübeck, Germany Heiko Hamann

Contents

Chapter 1
Introduction to Swarm Robotics

But it was one thing to release a population of virtual agents inside a computer's memory to solve a problem. It was another thing to set real agents free in the real world.

—Michael Crichton, Prey

The flying swarm is immediately sent into the 'cloud-brain' formation and its collective memory reawakens.

—Stanisław Lem, The Invincible

In fact, the colony is the real organism, not the individual.

—Daniel Suarez, Kill Decision

Abstract We introduce fundamental concepts of swarm robotics and get a little overview.

Swarm robotics is a complex approach that requires an understanding of how to define swarm behavior, whether there is a minimum size of swarms, what are the requirements and properties of swarm systems. We define self-organization and develop an understanding of feedback systems. Swarms do not necessarily need to be homogeneous but can consist of different types of robots making them heterogeneous. We also discus the interaction of robot swarms with human beings as a factor.

Swarm robotics is the study of how to make robots collaborate and collectively solve a task, that would otherwise be impossible to solve by a single individual of these robots. This true collaboration is the beauty of swarm robotics, it is the ideal of teamwork. Every swarm member contributes equally and everyone shares the same higher-level objectives. Public perception of swarm robotics seems to overemphasize the uncanny aspects though. Maybe you have read Stanisław Lem's *The Invincible*, Michael Crichton's *Prey*, Frank Schätzing's *The Swarm*, or Daniel Suarez' *Kill Decision*. Maybe you have wondered why most of these robot swarms or natural swarms are evil or at least threatening. It is unfortunate that these writers imagine our swarm future as a Dystopia (i.e., a frightening anti-utopia).

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2_1

An aspect that should not be left undiscussed but we leave it for later when we conclude the chapter with a discussion of future technology. Maybe you have also wondered whether the assumptions of Lem, Crichton, Schätzing, or Suarez are actually realistic. The main question is how can robot swarms be implemented and that is what we will study in this book.

While swarm robotics with its immediate application of collaborating robots gives you many robots that are at your service, it also helps to develop possibly even more important methods for the future of engineering. Technology of today has reached high levels. Among the most complex systems human beings have ever created are CERN's Large Hadron Collider and NASA's space shuttle. Other complex systems that have been engineered are CPUs (exponential increase of transistors per chip over time) and state-of-the-art (monolithic) robots [35, 244]. Common to these systems is that they are almost impossible to be understood by a single person because they contain many interacting components and many of these components are complex within themselves. To sustain command over the complexity of engineered products also in the future, we may reach a degree of complexity, when a paradigm shift in our methods is necessary [130]. Is there a method to generate and organize complex systems that does not rely on conquering a vast heterogeneous set of components and sub-components? Is it possible to generate complex systems based on simple components? Fortunately the answer is yes and we find examples of such systems in nature. Examples are swarms, flocks, and herds (see Fig. 1.1). Thousands of starlings self-organize in flocks and show collective motion. Shoals of fish move in circles forming big confusing clusters which make it difficult for predators to perceive individual fish. Locusts march in long aggregates through the desert and ants show the diversity of behaviors that is known by most of us. All these systems are composed of a high number of similar units. Each of these units are rather simple and have rather little capabilities relative to the complexity of the whole system that they form. These natural swarms embody the extraordinary concept of a complex system that is composed of simple components that interact based on simple rules. From an engineering point of view it would be appealing to know how to engineer such systems and from the scientific point of view it would be appealing to know how these systems operate. These two assignments are the two threads leading through this book and luckily they are directly interconnected. If you can build something, then you have really understood it; and if you have understood it, then you can build it.

Getting back to the quotations from Crichton and Lem at the top of this chapter, it will be an additional challenge to implement swarm behaviors not in simulated agents but in embodied agents, that is, robots. It will also be challenging to distinguish whether a certain observed behavior is a collective effect (e.g., a collective memory) or a mere individual reaction by a robot itself.

Besides the engineering challenge, there is also a much deeper challenge of uniting the two perspectives that each swarm necessarily has. There are features, that are only present on the swarm level, and there are features, that are only present on the level of an individual swarm member. As an observer we see the overall shape

Fig. 1.1 Swarms of honeybees (license Creative Commons, CC0), termites (photo by Bernard Dupont, CC BY-SA 2.0), a flock of geese (CC0), a flock of starlings (photo by Nir Smilga, CC BY 2.5), a school of fish (CC0), and a herd of sheep (CC0)

of the swarm, its dynamics, and we can often understand the purpose of each motion or the architectures that these swarms construct. For the individual robot, ant, or fish most of these features are hidden and yet it chooses the correct actions most of the time. To understand how the summation of simple behaviors results in qualitatively new features on the swarm level, that cannot possibly be understood by its parts, is the most intriguing problem of swarm intelligence and swarm robotics. Solving this riddle will have far reaching consequences because it is certainly related to neuroscience, sociology, and physics.

1.1 Initial Approach to Swarm Robotics

1.1.1 What Is a Swarm?

It is interesting to notice that there seem to be no explicit definitions of swarms in the literature. Common contributions, for example, from the field of swarm intelligence dare to keep a gap in this concern [48, 210]. Instead one finds definitions of swarming behavior. In biology a common definition of swarming behavior is "aggregation often combined with collective motion" which mostly agrees with our above examples. The importance of swarms as a concept is also communicated in the language itself. For example, in English there are words for a number of swarm behaviors, such as flocking in the case of birds, shoaling or schooling in the case of fish, and herding in the case of quadrupeds. Consequently, we owe a definition of a swarm and conclude for now that a swarm is defined via its behavior.

1.1.2 How Big Is a Swarm?

A question in swarm robotics that often raises a smile is the question for the appropriate size of a swarm. What seems to be clear is that it has to be many. But how many exactly? Fortunately in this case a definition by Beni [37] is found in the literature although he defines swarm size by what it is not: "not as large as to be dealt with statistical averages" and "not as small as to be dealt with as a few-body problem." Hence, for a proper swarm size N after Beni [37] we get $10^2 < N \ll 10^{23}$ which should be interpreted in the way that a swarm should not be "Avogadro-large." The Avogadro constant is $N_A \approx 6.02 \times 10^{23}$ 1/mol, whereas "mol" is the amount of substance containing as many elementary entities as in 12 g of pure carbon-12. The size of this upper bound might seem unreasonably high at first glance but it is useful because it has a qualitative meaning. Avogadro-large systems such as gases, liquids, or solids are typically treated statistically.

The lower bound is also defined method-relative as not to be dealt with as a "few-body" problem. If we investigate the complexity of few-body problems starting from small group sizes, then we can start from just one. A system with only one individual is certainly missing all interesting features, such as communication and collaboration. So we should go to two, which is the smallest possible social group, also called a dyad. The transition from one to two individuals changes the system fundamentally [38]. An example of a few-body problem is the two-body problem from classical mechanics. The problem is to determine the motion of two interacting bodies, such as a planet and its moon, based on Newtonian mechanics. Remarkably, the three-body problem shows already a qualitative difference (see Fig. 1.2) because it turns out to be a difficult mathematical problem [30]. Therefore, one might argue that the three-body problem belongs not to the class of simple few-body problems. Consequently the chosen lower bound of 10^2 might arguably be already too high while a lower bound of three might also be too extreme.

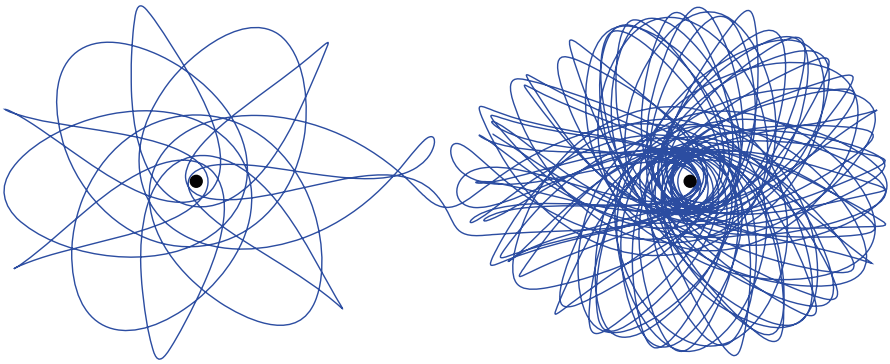


Fig. 1.2 Example trajectory from the three-body problem (Newtonian mechanics, two heavy stationary masses and a satellite in motion) as an example of complexity in "few-body" problems

Interestingly, this problem is related to the sorites paradox (sorites from soros, Greek for heap). It is the question about how many grains a heap must contain to be a heap. Sounds familiar? Although the question sounds silly, it is actually much deeper than one may think. One is tempted to solve the problem with a threshold (e.g., 10,000 grains), only noticing then that it is unfair to deny the 9,999-grain-aggregation being a heap. In the end it is a problem of vagueness in our natural languages that was discussed by such famous players as Russell [335].

The way out of this dilemma is to prevent setting thresholds and to stay with a method-relative definition. A swarm is not necessarily defined by its size but rather by its behavior. If it implements a swarm behavior then it should be considered a swarm and a swarm of $N = 3$ swarm members might show a verifiable swarm behavior. Properties and requirements of swarm behaviors are discussed in the following.

1.1.3 What Is Swarm Robotics?

First implementations of swarm robots go back to Maja J. Matarić and her "Nerd Herd," see Matarić [253, 254, 255]. A definition following Dorigo and Şahin [99] is: "Swarm robotics is the study of how a large number of relatively simple physically embodied agents[1] can be designed such that a desired collective behavior[2] emerges from the local interactions among agents and between the agents and the

[1]The concept of "agents" is well-defined and complex, see Russell and Norvig [337] and Sect. 2.4.2.

[2]"Collective behavior" is a technical term common in sociology which is used in swarm robotics research rather ingenuously. It seems sufficient to read it with the simple meaning "behavior of the whole swarm," that is, the resulting overall behavior of all swarm members.

environment." The term "relatively simple" agents can furthermore be specified task-relatively as relatively incapable or inefficient robots relative to the considered task. The fact that local interactions between agents and the environment should be possible requires robots to have local sensing and probably also communication capabilities. In fact, (local) communication is often considered a key feature of swarms [54]. Communication is also a necessary requirement to allow for collaboration and cooperation between swarm members. Collaboration, in turn, is required to go beyond a mere parallelization in a swarm system. In the simplest form of parallelization, for example, each processor receives a workpackage that is then processed without any other required communication between processors until the workpackage is completed. Although this simple form of parallelization is an option for robot groups (imagine a cleaning task with each robot individually cleaning a small, assigned area), in swarm robotics we want to clearly go beyond that concept and generate an additional performance boost by collaboration between robots.

Beni [37] defines an intelligent swarm as "a group of non-intelligent robots forming, as a group, an intelligent robot. In other words, a group of 'machines' capable of forming 'ordered' material patterns 'unpredictably'." This definition requires however many other involved concepts such as intelligence, order, and unpredictability and is not pursued here. Instead we note his definition of properties of a robot swarm: decentralized control, lack of synchronicity, simple (quasi) identical members or quasi-homogeneous members which are mass produced [37]. Decentralized control is the opposite of centralized control which is the control of remote units by a central unit. Decentralized control, hence, is the control paradigm that prohibits the control of units by other units which means that each unit controls itself. The lack of synchronicity means that the swarm is asynchronous which is the lack of a global clock to which each swarm member could have access. In turn, a swarm has to explicitly synchronize if necessary.

Sharkey [359] discusses different types of swarm robotics. She argues that early approaches of swarm robotics were nature-inspired and "minimalist approaches" that limited the capabilities primarily of the hardware which seems not to be a consensus in the community anymore. Hence, she identifies *scalable swarm robotics* which is not minimalist and not directly nature-inspired, *practical minimalist swarm robotics* which is not directly nature-inspired, and *nature-inspired minimalist swarm robotics* which is, if you like, the original.

1.1.4 Why Swarm Robotics?

When benefits of swarm robotics systems are discussed, typically three main advantages are mentioned [99]. The first advantage is that these systems are robust. Robustness is defined as fault tolerance and fail-safety and it is achieved by massive redundancy and the avoidance of single-point-of-failures. A swarm is quasi-homogeneous and hence each robot can be substituted by any other.

Note that redundancy also requires a certain minimal swarm size. Above we have mentioned that a group of size three could be considered a swarm depending on the methodology. That is still a fair assessment but it is also clear that a swarm of size three achieves not the same level of fail-safety as a swarm of size 100.

The control of the swarm is decentralized and therefore a single-point-of-failure does not exist. Every robot interacts only with direct neighbors and it also stores only locally obtained data (i.e., local information). Losing a robot due to any kind of failure consequently has only a local effect which is easily overcome. In an appropriately organized robot swarm even a certain percentage of robots may be lost without big effects on its effectivity. The efficiency might drop but the given task is still solved completely.

We give an example for a more detailed understanding of robustness. A network of swarm robots can be interpreted graph-theoretically. Say for the given task it is essential that the swarm stays connected in one large connected component. A situation that is to be avoided in order to guarantee robustness is that the removal of one edge results in a separation of the graph into two components. The explicit determination of whether a certain link is crucial is non-trivial and difficult in the given decentralized setting. An easy solution would be to guarantee a certain minimal robot density (i.e., number of robots per area) that implies a certain average degree (i.e., number of neighboring robots in communication range) of each robot. This is only a probabilistic guarantee which is, however, typical for swarm robotics systems. Robustness can be considered in graphs. In Fig. 1.3 node A and edge b represent single point of failures. If A or b would fail, then the graph would be separated in several connected components (cf. vertex separator in graph theory). The graph can represent a communication network formed by swarm robots and their communication range. Initially, each robot can reach each other robot either explicitly via multi-hop communication or indirectly by diffusion of information through the network (e.g., sequences of local broadcasts). Once robot A fails or communication along edge b is not possible anymore, the robot swarm collapses into two subpopulations that cannot communicate with each other anymore.

A good example for robustness by redundancy is the following. In today's space flight typically single space probes are launched that constitute a central approach (see Fig. 1.4). Whenever a fatal failure occurs in this probe the whole mission fails. Hence, no effort is spared to perfect these probes and to test them in any possible

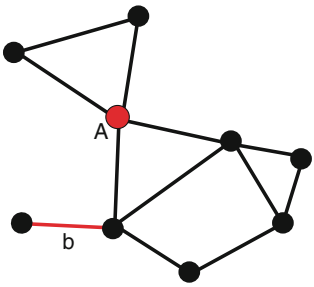


Fig. 1.3 In this graph, node A and edge b represent single point of failures because if they fail, the graph is separated into several components

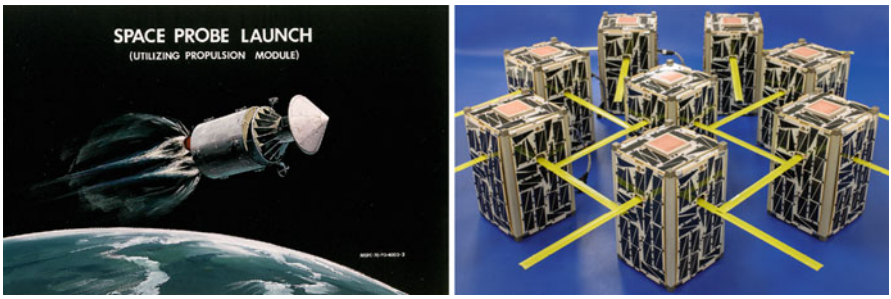


Fig. 1.4 Robustness by redundancy: instead of launching a single space probe (left), one can launch a group of probes or satellites (right, project EDSN—Edison Demonstration of Smallsat Networks) (photos by NASA, public domain)

way. An alternative would be to produce a high number of small and possibly simple probes, in the ideal case for the same amount of money. The limited capabilities of each individual swarm probe (e.g., radio signal strength, resolution of camera) is compensated by cooperation. Losing a certain percentage of swarm probes would not endanger the whole project because the probe swarm stays effective. A robust system is a reliable system and a reliable system can be created based on unreliable parts via redundancy [29].

The second advantage is flexibility. Again due to quasi-homogeneity there is no specialization in terms of hardware. Each robot is able to complete the task of any other. The swarm is able to adapt to a wide range of tasks because often specialization in terms of hardware is not necessary. The robots overcome limits in their capabilities by cooperation. For example, too weak actuators are compensated by the collective transport of objects, too low signal intensities are compensated by forming multi-hop communication lines of robots, and too small dimensions are compensated by self-assembling robots that collectively cross holes or other difficult obstacles.

The third advantage is scalability. The applied methods such as the control algorithms of the robots scale to any size of the swarm. It "can maintain its function while increasing its size without the need to redefine the way its parts interact" [96]. This is because each robot interacts only with its local neighborhood. Methods that do not scale, such as broadcasts to the whole swarm or processes that depend on surveying large fractions of the swarm, are prohibited. For scalability the guarantee of a constant robot density might be necessary but the overall swarm size is allowed to be chosen freely when the area in which the swarm operates is scaled up accordingly.

An example for the limits of scalability in classical computer systems are Internet services. An increasing number of users is answered by an increasing number of servers. The difficulty is that typically some bottleneck exists. For example, incoming service requests might need to be registered at a central machine. As a consequence response times do not stay constant with increasing number of users and they also do not scale linearly but exponentially.

1.1.5 What Is Not Swarm Robotics?

Sometimes multi-agent or multi-robot systems are difficult to separate conceptually from swarm robotics systems. Multi-agent systems, however, often rely on the distribution of or access to global information, broadcasts, sophisticated communication protocols (e.g., negotiations) that require reliable robot–robot communication, explicit assignments of roles that require robots to identify individual robots or to know the total swarm size [117, 298]. These systems typically do not scale also because of applied, non-scalable technologies such as bluetooth or WLAN (wireless local area network, aka WiFi).

1.2 Early Investigations and Insights

Next, we discuss three essential and fundamental features of swarm robotics systems. We start with a detailed discussion of swarm performance, that is, how well the swarm is doing depending on the swarm size and the swarm density. As we hope for good scalability, studying swarm performance is important. As second subject we discuss communication because swarms sometimes communicate in unexpected ways. As third subject we specify the concepts of the swarm level and the level of the individual swarm member.

1.2.1 Swarm Performance

The average performance of a robot swarm depends on the swarm density or the swarm size if the area in which the swarm operates is kept constant. This is readily visualized by the example of bucket brigades. A bucket brigade is a multistage partitioned transport scheme which can be seen as a form of task partitioning [8]. Forming a bucket brigade is useful once the swarm can cover most of the distance between source and sink. Bucket brigades also occur in a number of ant species [8]. When there is only one robot then it has to pick up water at the one end and transport it to the other end by driving there. Similarly, for low numbers of robots they have to go back and forth, whereas two generate a water flow double that of a single robot, four double that of two and so on (see Fig. 1.5a).

Say, once there are six robots they are able to form a bucket brigade which means they stay fixed in their position and transport water only by handing it over to the next in the brigade. By this simple form of cooperation these six robots are able to transport even more than double the amount of water compared to what three robots can do (see Fig. 1.5b). This constitutes a superlinear performance increase which is indicated by a big jump in Fig. 1.5c. A superlinear performance increase means that not only the overall performance increases with increasing team size, also

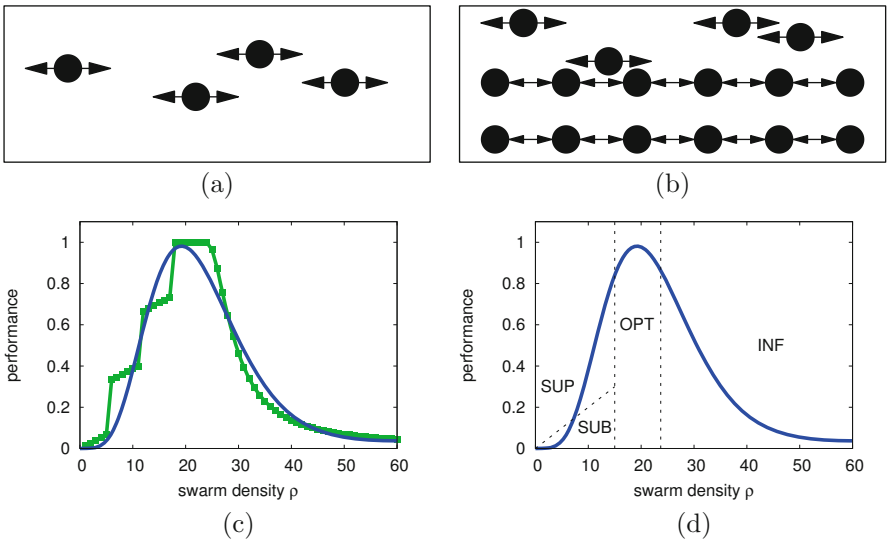


Fig. 1.5 Bucket brigade example for swarm performance (robots have to transport objects back and forth between the left and right side of the robot arena) and typical swarm performance function over swarm density $\rho = N/A$ for a fixed area $A = 1$ (without units). (**a**) Bucket brigade, $N = 4$ robots (**b**) Bucket brigade, $N = 16$ robots (**c**) Bucket brigade, performance. (**d**) Swarm performance showing four regions, SUP: super-linear, SUB: sub-linear, OPT: optimal, INF: inference

the individual performance increases with increasing team size. Imagine you are studying in a group of students and by increasing the group size everyone of you gets more efficient. That is a great and very desirable effect but unfortunately only rarely observed in teams. Increasing individual performance with increasing swarm size was observed, for example, in wasps [198] and it is also observable in swarm robotics as we will see later (Chap. 4).

When more and more robots are added to the system the robot density continues to increase and for a certain robot density an optimal swarm performance is reached. When the robot density is further increased then the swarm performance actually decreases because robots interfere with each other and slow themselves down. When the robot density is increased even further then the performance continues to drop possibly down close to zero when almost all motion is stopped due to interference [236]. Even ants have to deal with physical interference sometimes although they have fantastic crawling and climbing abilities. In crowded conditions they avoid traffic jams by negative feedback effects [107].

Besides physical interference also other effects can increase the overhead due to high swarm densities. An example from the performance of human groups is the so-called Ringelmann effect which describes the loss of motivation of individuals working in groups with increasing group size. A nonlinear decrease of individual performance with increasing group size is reported by Ingham et al. [193] (see also [210, p. 236]).

We divide a robot swarm's performance qualitatively into four regions: sublinear increase, superlinear increase, optimal swarm density (could also be called the operating point), and decrease due to interference (see Fig. 1.5d). The shape of the function shown in Fig. 1.5d occurs in many different swarm systems [34, 164, 171, 182, 216, 227, 236] such that it seems justified to define it as a general scheme for swarm performance [166]. The existence of such a general shape is also reported by Østergaard et al. [299] in the context of multi-robot systems:

> We know that by varying the implementation of a given task, we can move the point of "maximum performance" and we can change the shapes of the curve on either side of it, but we cannot change the general shape of the graph.

We define the general shape of swarm performance Π depending on the swarm density ρ (robots per area), a fixed area A, and swarm size $N = \rho A$ by

$$\Pi(N) = \frac{N}{e^N}, \tag{1.1}$$

which is roughly $0.37N^{1-N}$. With all relevant constants we get

$$\Pi(N) = C(N)(I(N) - d) \tag{1.2}$$

$$= a_1 N^b a_2 \exp(cN), \tag{1.3}$$

for parameters $c < 0$, $a_1, a_2 > 0$, $b > 0$, and $d \geq 0$ [166]. Parameter d is used to set the limit $\Pi(N \to \infty) = I(N \to \infty) - d = 0$. Furthermore, we refine our definition of $\Pi(N)$ by introducing two components, C and I. We define a theoretical swarm effort that could be reached without any negative feedback by the cooperation function

$$C(N) = a_1 N^b. \tag{1.4}$$

The same formula was used by Breder [55] in the context of cohesiveness within fish schools and by Bjerknes and Winfield [42] to model swarm velocity in emergent taxis. They used parameters of $b < 1$ while we also allow $b > 1$. Potentially that is a major difference because $b < 1$ represents a sublinear performance increase due to cooperation while $b > 1$ represents a superlinear increase. We define the interference function by

$$I(N) = a_2 \exp(cN) + d. \tag{1.5}$$

This can be interpreted as the theoretical swarm performance that could be achieved without any cooperation (i.e., without positive feedback). Nonlinear effects that decrease efficiency with increasing swarm size are plausible due to negative feedback processes such as the collision avoidance behavior of one robot that triggers the collision avoidance behavior of several others in high density situations. Still, there are many options of available nonlinear functions but best results were

obtained with exponential functions. Also Lerman and Galstyan [236, Fig. 10b] report an exponentially decreasing efficiency per robot in a foraging task.

There is a model for parallel processing performance in distributed systems by Gunther [158]. He calls it the Universal Scalability Law. For a relative capacity $R(N)$ (i.e., performance) we define

$$R(N) = \frac{N}{1 + \alpha((N-1) + \beta N(N-1))}, \tag{1.6}$$

for a coefficient α that gives the degree of contention (inference) in the system and coefficient β that gives the lack of coherency in the distributed data. Contention occurs because resources are shared. Whenever the capacity of a shared resource is used completely and another process requests to use that resource, then the process has to wait. Contention increases with increasing system size, while keeping resources at the same capacity. Lack of coherency occurs because processes, to a certain extent, operate locally. For example, they have local changes in their caches that are not immediately communicated to all other processes. Maintaining coherency is costly and the costs increase with increasing system size.

Gunther identifies four qualitatively different situations:

a. If contention and lack of coherency are negligible, then we get "equal bang for the buck" and have a linear speedup ($\alpha = 0$, $\beta = 0$, Fig. 1.6a).
b. If there is a cost for sharing resources in the form of contention, then we have a sublinear speedup ($\alpha > 0$, $\beta = 0$, Fig. 1.6b).
c. If there is an increased negative influence due to contention, then the speedup clearly levels off ($\alpha \gg 0$, $\beta = 0$, Fig. 1.6c).
d. If in addition there is also an increased influence of incoherency, then there exists a peak speedup and for bigger system sizes the speedup decreases ($\alpha \gg 0$, $\beta > 0$, Fig. 1.6d).

In the original work of Gunther [158], superlinear performance increases are basically not allowed. In a more recent work [159], superlinear speedups are discussed and negative contention coefficients $\alpha < 0$ are allowed now (see Fig. 1.6e). While contention $\alpha > 0$ refers to capacity consumption due to sublinear scalability, $\alpha < 0$ refers to a capacity boost due to superlinear scalability. In parallel computing, superlinear speedups can occur due to some interplay between problem size per computing unit and available memory. For example, if the problem can be divided into pieces that fit completely into a CPU's cache, then one can observe a considerable speedup. In swarm robotics, superlinear performance increases occur due to qualitatively different collaboration modes that are accessible with increasing swarm size as seen in the bucket brigade example above or when assembled swarm robots cross a hole in a team.

In the context of swarm robotics we can interpret contention as interference between robots due to shared resources, such as an entrance to a base station or generally space. Following this interpretation, collision avoidance is a waiting

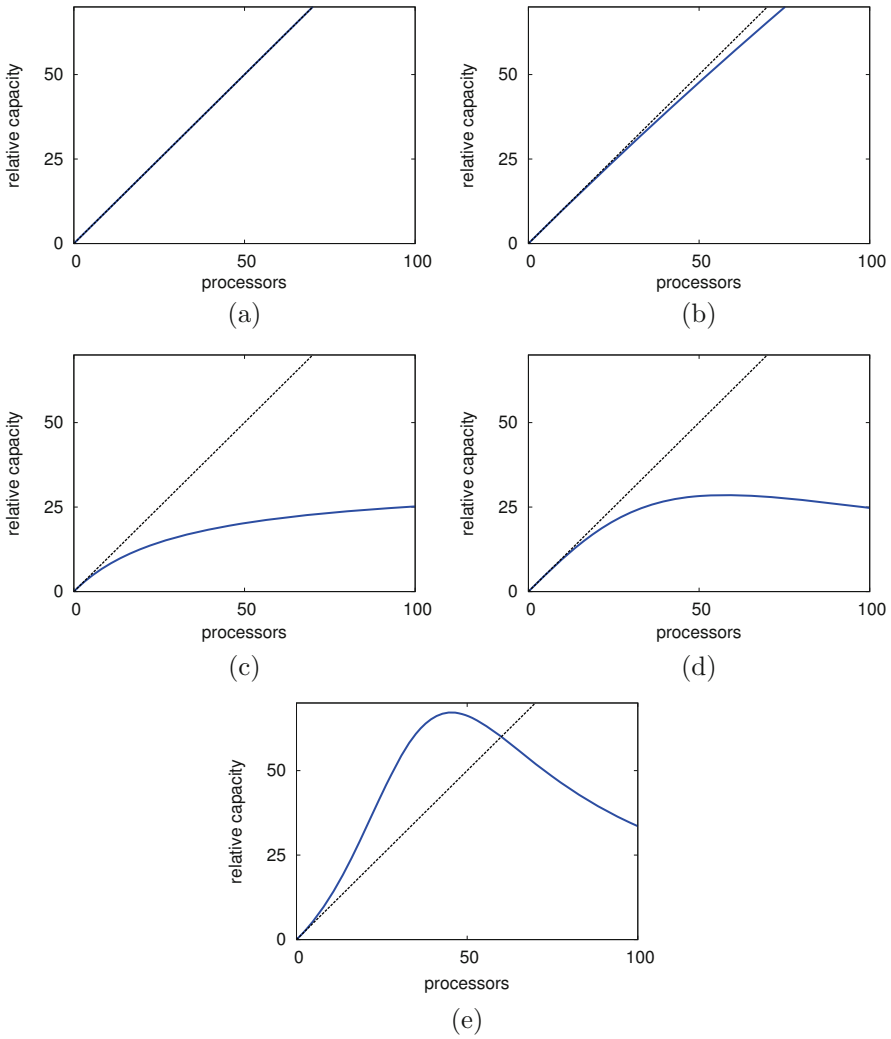


Fig. 1.6 Universal Scalability Law following Gunther [158], four standard situations and superlinear speedup [159] depending on parameters α (degree of contention) and β (lack of coherency). (**a**) Linear speedup, $\alpha = 0$, $\beta = 0$ (**b**) Sublinear speedup, $\alpha = 0.001$, $\beta = 0$ (**c**) Speedup levels off, $\alpha = 0.03$, $\beta = 0$ (**d**) Decrease, $\alpha = 7{\times}10^{-4}$, $\beta = 3{\times}10^{-4}$ (**e**) Superlinear, $\alpha = -0.03$, $\beta = 5{\times}10^{-4}$

loop because the shared resource *space* is currently not available. That is intuitive and similar to an airplane flying a holding pattern because the resource *runway* is currently in use and should certainly not be shared. Incoherency, in turn, can be interpreted as inconsistencies or overhead due to limited communication of information or due to imperfect synchrony.

While Gunther assumes that there cannot be a system-wide deadlock situation due to contention (speedup monotonically increases with increasing α), that could occur in a swarm robotics system. For example, the swarm density could be too high, such that all robots permanently try to avoid collisions resulting in zero performance.

1.2.2 Communication

As discussed above, communication is essential to allow for collaboration in a swarm. In swarm robotics we distinguish between two kinds of communication: explicit and implicit communication. Explicit communication is the classical concept of communication, that is, there is a communication channel between sender and receiver and an explicit message is sent. Implicit communication does not have an explicit communication channel nor an explicit message. It is also called cue-based communication. For example, by merely perceiving the presence of another robot, a robot might adapt its behavior accordingly. Hence, one can say that a robot communicated implicitly with the other one. In biology there are two classes of interaction between animals at a distance. First, a cue is an unintentional index. For example, a trail in the snow is left as a cue by one animal which can then be exploited by another animal. For example, it can choose to follow the trail if it is hunting for prey or it chooses to avoid the trail if it wants to avoid approaching it. Second, a signal is an intentional index. For example, an alarm cry of a bird is an intentional warning to peers.

Another relevant communication concept is that of stigmergy which is a form of cue-based communication [151]. It is the communication by way of the environment. An example are pheromone trails in ants. An ant drops pheromone on the ground which is an information for ants that pass this position later. Hence it can be seen as a form of delayed communication. It is like using the environment as a blackboard where you can leave and edit messages for whoever may pass by later. Other examples of stigmergy operate on other materials that are not exclusively used for communication. For example, building material in nest construction serves as indicator where to continue to add material or in other situations aggregations of objects serve as messages.

1.2.3 Two Levels: Micro and Macro

Naturally swarm systems consist of two levels. First, there is a microscopic level (also called local level) which is the level of the individual robot, the level which is the local view dominated by limited knowledge, limited sight distance, uncertainty, and primitive actions. Second, there is a macroscopic level which is the level of the whole swarm, the level which is the global view dominated by full knowledge,

Fig. 1.7 Two levels, micro and macro, figuratively microscope (Binocular compound microscope, Carl Zeiss Jena, 1914) and telescope (photo by Azmie Kasmy, CC BY-SA 3.0)

full overview, at which the overall task is defined. Often also the terms "local" for microscopic and "global" for macroscopic are used. As shown in Fig. 1.7, the micro and macro level figuratively correspond to the microscope used to explore the microscopic worlds (e.g., intracellular communication) and the telescope used to explore the macroscopic worlds (e.g., dynamics of galaxies).

The existence of these two levels implies a grand challenge for the design of swarm robotics systems. The task of the swarm is defined on the macro-level (swarm behavior) while the implementation of the robot controller is to be done on the micro-level (executable code for the robot's microcontroller).

Furthermore there is also a classification of swarm models based on these two terms. Macroscopic models abstract away microscopic details, that is, they do not model all properties of individual robots (e.g., their position, internal state). Microscopic models, in turn, represent each robot explicitly with all necessary details (e.g., full trajectories of all robots, a robot's state of its internal memory). Sometimes researchers distinguish modeling approaches in Lagrangian and Eulerian [302]. Lagragian refers to microscopic models that represent properties of each individual explicitly (e.g., velocities and internal states). Eulerian refers to macroscopic models that represent only group properties in continuum equations (e.g., density of the swarm with partial differential equations).

Sometimes it is difficult to understand and accept that there really is a macroscopic level in the system. Often one can break down seemingly macroscopic features into smaller pieces in a reductionist approach and hence macro-effects vanish. However, there are studies that address macroscopic concepts, such as collective perception [347, 393] and collective memory [78]. In collective perception an object or environmental feature is perceived and recognized not by the individual robot but by the swarm as a whole. Each individual robot has only access to a limited

amount of information (local information) that is not enough to infer the overall property of the investigated object or the global feature of the environment. The information is strictly distributed within the swarm and only the whole swarm can perceive and create awareness. Similarly, a collective memory can emerge "where the previous history of group structure influences the collective behaviour exhibited as individual interactions change" [78]. Again, the respective information cannot be localized within the swarm but instead is distributed over all swarm members on the macroscopic level. The "cloud-brain" mentioned in the quotation of Lem at the beginning of this chapter would be another example of a macroscopic feature in the form of collective memory. Still, any particular operation within this memory could be broken down into operations within an individual robot, just as one can do for a regular computer memory. However, the system is only fully grasped when we accept macro-features and when we look at the big picture. This way the macroscopic level would be only epistemological (i.e., a model assumption that helps to understand the system but is not necessarily physically existent), such that one could still argue that it is not really existent. Yet, that is a philosophical discussion that would go beyond the scope of this book.

1.3 Self-Organization, Feedbacks, and Emergence

An essential concept in swarm robotics is self-organization. All robots act based on local perception while the swarm is supposed to complete a task that is defined on the macroscopic level. The idea of self-organization explains exactly this relation between micro and macro and gives an answer to the question how macro-structures can be generated based on micro-interactions only. In a populist but concise statement self-organization can be described as a generator of "order from noise." It describes how spatial, temporal, and spatiotemporal patterns emerge in open systems driven away from thermal equilibrium. Open systems permanently "consume" energy and therefore they are able to generate structures. We cannot create order without wasting energy as described by the second law of thermodynamics. Animals have a metabolism and digest food that contains chemically bound energy. Robots have batteries that provide power to run the motors, CPU, sensors, and actuators.

Self-organization is based on four components: positive feedback, negative feedback, multiple interactions, and balance of exploitation and exploration [48]. Feedbacks introduce dynamical non-linearities to the system which are necessary to generate complex behaviors. Multiple interactions occur because self-organizing systems consist of many entities that interact by physical contact or by means of direct or indirect communication. The historic origin of self-organization are systems known from the field of physics, that is, non-living systems [162]. For example, the formation of a crystal transforms a gas or a liquid into solid form. An initial fluctuation (random event) forms a crystal nucleus which is then reinforced by the crystal growth. Hence, the above statement "order from noise" applies here. The form of the crystal is to a large extent determined by the underlying atomic

and molecular characteristics of the material. Still, there is room for variety as known from the phenomenon of snowflakes. The balance between exploitation and exploration in a self-organizing system makes it adaptive. Even if repeating a certain action seems the only profitable strategy for now (i.e., exploitation), it makes sense to keep exploring, for example, the environment and keep checking for changes. Only if the swarm learns about these changes it can adapt to them and switch its behavior. An example is an ant colony that found a great food source but still keeps exploring alternative food sources. The currently exploited food source may deplete or a predator may increase the cost of getting there.

The most important two of the four components of self-organization are the feedbacks. They provide a concrete and rich schema to identify and interpret self-organizing systems. Next we study positive and negative feedbacks in more detail.

1.3.1 Feedbacks

Feedbacks are a seemingly simple concept but not always easy to identify and sometimes it is difficult to distinguish positive from negative feedback in a self-organizing system. In the following we restrict ourselves to a simple definition and later in the book we try to identify feedbacks in systems of swarm robotics.

The effect of positive feedback is a force that increases, inflates, and escalates quantities. In self-organization positive feedback is important because it breaks the initial randomly uniform distribution of system properties (homogeneity, without irregularities). A fluctuation creates initially small deviations that are then reinforced by positive feedback. Examples are the snowball effect, path formation, and a bubble at a financial market. They require only a small initial trigger, a snowflake starting to roll downhill, a human being walking across an unmown hayfield, or rumors about the future profitability of Internet companies, which can then potentially escalate to a process that makes a difference. The economic view[3] is that an individual

> may think there is an opportunity for speculation: to buy the stock now and sell to other buyers at a profit later. [...] the initial increase in demand creates a positive feedback leading to further increases in demand.

Another example is the insane requirement of permanent growth in capitalism to allow for payment of interests which requires economies to grow exponentially. Unfortunately, our earthly resources are limited which means that even processes that are driven by positive feedback need to stop at some time. Even the biggest snowball rolls to a stop once it has reached the bottom. A path across a hayfield will not attract an infinite amount of people. Every bubble ends in a stock market crash. Another positive feedback process is the so-called Matthew effect from sociology. In reference to the Gospel of Matthew, it is the effect of the increasing gap between

[3]Open-access book "The Economy," http://www.core-econ.org/the-economy/book/text/11.html#118-modelling-bubbles-and-crashes.

rich and poor: the rich get richer and the poor get poorer. This effect is, for example, observed in the creative industries, where small initial advantages in the sales figures of a medium generate attention and hence promote further sales. In swarm robotics we can make use of positive feedback, for example, in collective decision-making where we want to see a decision on the global level in the form of a consensus (all swarm members agree on one option).

A system that has only positive feedback will explode because there is no limit, that is, it will generate snowballs of infinite size or infinite stock prices. Every system operating on finite resources, hence has also negative feedback components. Negative feedbacks stop positive feedback processes due to limited resources. For example, a positive feedback process that reinforces the majority in opinion dynamics is finally stopped because we are running out of voters that are of opposite opinion. The effect of negative feedback is a force that reduces deviations, diminishes majorities, and damps overshoots. Examples are control systems such as the centrifugal governor or the baroreflex which keeps the blood pressure constant. An increased blood pressure effects a reduction of the heart frequency which then decreases the blood pressure. A too low blood pressure, in turn, inhibits the baroreflex which then increases the blood pressure. The centrifugal governor is a nice piece of engineering that can be used to regulate the rotational speed of a steam engine (see Fig. 1.8). The centrifugal force lifts up masses that in turn regulate the intake of steam via a valve. In swarm robotics we can make use of negative feedback, for example, to keep the swarm in a certain equilibrium state or to keep the swarm away from too extreme system states.

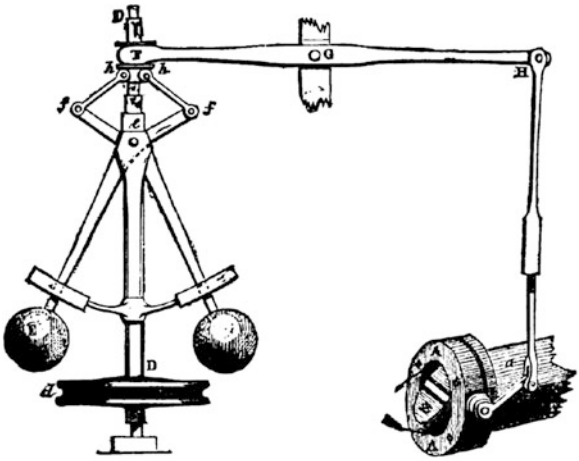


Fig. 1.8 Centrifugal governor as example of negative feedback (public domain)

1.3.2 Examples of Self-Organizing Systems

A dynamic example is Rayleigh-Benard convection. When a relatively flat layer of a fluid (e.g., oil or water) is heated homogeneously from below, flows of warm and cool fluid are generated. Based on initial fluctuations these flows self-organize into convection cells with an upflow of warm fluid in one half of the cell and a downflow of cold fluid in the other half.

Another example is that of the laser (acronym for light amplification by stimulated emission of radiation) which is maybe counterintuitive. Laser light is a coherent light wave that is created by "pumping" energy into a gain medium (e.g., a gas). From the point of view of self-organization, the atoms in the medium synchronize and emit light in a coordinated way. Hermann Haken contributed a theory of lasers which lead to the development of a theory called "synergetics" [161, 162]. Synergetics is basically a clever method to reduce systems described by multiple differential equations down to a few differential equations while still representing the essential system components. Synergetics can also be seen as a theory of self-organization although it seems not to have high potential for generalization.

A third phenomenon of interested is self-organized criticality which is, for example, shown in sand piles of increasing size [24]. When a constant flow of sand is added to a pile, the pile increases in its height and also the sides are getting steeper (see Fig. 1.9). The friction between grains functions as a stabilizing force which is counteracted by gravity that implies an increasing force due to an increasing weight of the pile. In the moment when the force by gravity equals the force by friction the system is critical. When even more sand is added, this criticality is released by an avalanche that decreases the steepness of the pile. An interesting characteristic of this system is the distribution of the sizes of avalanches which is a power-law distribution. Small avalanches are exponentially more likely than big avalanches which have, however, non-zero probabilities. That is why one speaks of a long-tail distribution. To summarize we note that energy is added to the system by placing sand grains on top (potential energy), multiple interactions are the physical contacts between neighboring grains, and order is shown in the

Fig. 1.9 Sand pile as example of self-organized criticality

Fig. 1.10 Examples of natural pattern formation: pigmentation of fur and sea shells (photos CC0)

power-law distributed avalanches, that is, temporally separated events in contrast to the constant flow of added sand.

The behavior of a self-organizing system can also be simple as seen in systems showing the Brazil nut effect [155, 280]. This is a simple experiment which might entertain the reader while having breakfast if cereals are preferred. Say a box contains a granular medium such as nuts or balls of different sizes. Energy is added to the box by shaking it. Observed is a sorting effect: big grains (or nuts, balls, etc.) end up on top of the box and small grains are found at the bottom. This effect is generated by small grains filling the gaps beneath bigger grains while these are jumping up due to the shaking process.

Examples of self-organization in living systems are pattern formation and behaviors in insects. Pattern formation is found in developmental processes in animals such as embryology [144, 157, 195, 205, 424], the pigmentation of fur [60, 289], and the pigmentation of sea shells [264, 266, 267] (see Fig. 1.10). The multitude of interactions is implemented by communicating cells and energy is added via the cells' metabolism. Social insects often show complex behaviors, for example, in foraging or nest building. It seems possible that some simple behaviors in these animals are genetically programmed, that is, they might be hardwired in their brains. It seems, however, unlikely that this is true for many of their behaviors. A more plausible explanation seems that evolution selects behavioral rules that capitalize on principles of self-organization [122]. These simple rules generate the necessary behavior not directly but via interactions between individuals that implement a self-organizing process. This way the required behavioral complexity is outsourced to some extent into the multiple interactions between many individuals sharing similar genetic material.

Self-organization is by some authors even linked to such prominent research questions as that for the origin of life. The chemical origin of life might be based on hypercycles and autocatalytic systems [110, 111]. The evolution of non-cellular life may have been based on adsorption on grains. The evolution of cellular life, in turn, maybe was based on self-assembly of viruses. Other examples of self-organizing systems are percolation [153] and diffusion-limited aggregation [421]. Percolation is a phenomenon that is observed in a diverse set of scenarios, such as forest fires and

filtering of fluids through porous materials. In a forest, the density of trees influences whether the fire may eat through the whole forest or stop before. Similarly, the density of holes in a porous material determines its permeability. Diffusion-limited aggregation is basically an aggregation process of particles approaching a seed from many different directions in random walks. Interesting tree structures are formed this way.

1.3.3 Emergence

Emergence is a difficult to define and vague philosophical concept. Still, it is of interest for swarm robotics because it might be a promising approach for designing robust and complex systems. It is the idea that totally new properties might emerge on a higher level that cannot be described by concepts of the lower level (cf. also holism and reductionism). Many definitions of emergence use concepts in dependence on Aristotle's "the whole is greater than the sum of its parts," surprise of an observer, fundamental novelty, or unpredictability [1, 47, 83, 88, 89, 223, 224, 411]. Holland [190] just throws up his hands in defeat: "It is unlikely that a topic as complicated as emergence will submit meekly to a concise definition, and I have no such definition to offer." Johnson [200] discusses the amazing observation of maturing in ant colonies that takes longer than the life of an individual ant: "How does the whole develop a life cycle when the parts are so short-lived? It would not be wrong to say that understanding emergence begins with unraveling this puzzle." Concerning the prediction of emergent systems Johnson [200] refers to Gerald Edelman's experience: "You never really know what lies on the other end of a phase transition until you press play and find out. That is the lesson of Gerald Edelman's recipe for simulating a flesh-and-blood organism: you set up a system of various pattern-recognition devices and feedback loops, connecting the virtual organism to a simulated environment. And then you see what happens." It is that potential unpredictability of such systems that complicates the design of control algorithms for swarm robots.

Researchers from the field of swarm robotics have addressed the problem of defining emergence as well. Bayindir and Şahin [33] state: "Emergence is a key property in complex systems which means the behavior of the complex system cannot be understood by examining only the components of the system. Although the components of the complex system can be simple, the resultant system may be complex because of the interactions of the system components." Following Beni [37] and Şahin [338], emergence can also serve as an indicator that a particular control algorithm is consistent with principles of swarm robotics. Bjerknes et al. [43] claim: "The algorithm meets the criteria for swarm robotics [...] We have a highly robust and scalable swarm of homogeneous and relatively incapable robots with only local sensing and communication capabilities, in which the required swarm behaviors are truly emergent." Dorigo et al. [97] have a simple definition for ant systems to offer: "Solutions to problems faced by a colony are emergent rather than predefined."

The philosophical concept of emergence goes back to John Stuart Mill [276, 363] and is described as the "concept of genuinely new kinds of properties produced by nature that cannot be reduced" [352]. An underlying assumption is that there are irreducible properties, for example, in a way that "psychology is not applied biology, nor is biology applied chemistry" [10]. The importance of interactions among many entities was stressed by Prigogine [317]: "As long as we consider merely a few particles, we cannot say if they form a liquid or gas." But once there is many of these particles we can observe them in different phases (e.g., liquid or gaseous): "Phase transitions correspond to emerging properties" [317]. The question whether true novelty of these emerging properties can really exist is again rather philosophical [80, 81]. Prigogine [317] is positive and summarizes his position in poetic language: "Nature is indeed related to the creation of unpredictable novelty, where the possible is richer than the real."

From the programmer's point of view in swarm robotics, emergence can be understood as "programming between the lines." The local behavior of an individual robot is defined by the programmer via the robot's control program without explicitly defining the emergent, global behavior which cannot be perceived by an individual robot. Emergence is an intriguing concept because it constitutes the engineer's dream—the construction of complex systems by an unproportionally small effort.

1.4 Other Sources of Inspiration

Following Şahin [338] there are additional sources of inspiration in addition to the above examples of self-organization. Also Baluška and Levin [28] remind us in their article "On having no head" that cognition does not require necessarily a brain (see Fig. 1.11). Even tissue cells, the cytoskeleton (filaments in the cell that can contract and influence cell migration), and the underlying genetic network can sense environmental changes and react to them. An important source of inspiration is, for example, the slime mold [3]. The slime mold is an amoeba that is able to operate in three modes of operation: a single amoeba, a swarm of amoebas in collective motion, and an aggregate of amoebas forming a pseudo-organism. A slime mold

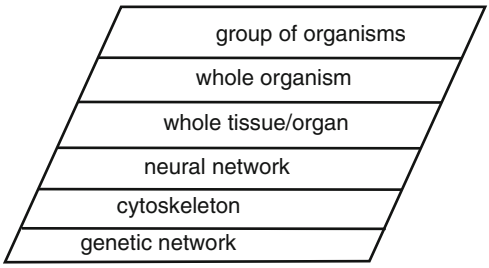


Fig. 1.11 Cognition on different levels in biological systems [28]

amoeba excretes a chemical substance called cAMP at small portions periodically. If the local environment of the amoeba exceeds a given threshold, the amoeba excretes a much higher amount of cAMP molecules (positive feedback). Amoebas walk uphill in their local cAMP field (i.e., towards higher concentrations of cAMP). A trail-like formation towards the center of an aggregation forms. Finally a snail-like pseudo-organism is formed with up to 10^5 cells, which is called the "slug" state of a slime mold.

That way the slime mold shows modes of operation with different degrees of coupling between amoebas. With a single, isolated amoeba there is no coupling. The collective motion of amoebas is a typical swarm mode with moderate coupling. The pseudo-organism mode is characterized by an intensive, physical coupling that goes beyond the typical cooperation in swarms. Temporarily such an intensive coupling is, however, also found in swarms. For example, ants are known to self-assemble themselves into chains of physically connected ants to form bridges between leaves or over small ditches [138].

Another source of inspiration is quorum sensing in bacteria. Similarly to the slime molds, bacteria also show intelligent behavior based on interactions. It is important to note that these unicellular organisms control themselves by different means compared to vertebrates that utilize nerve cells. The behavior of unicellular organisms is generated by intracellular communication that relies on intracellular signals, diffusion processes, and sophisticated processes that implement the interaction between the cell and its environment through its membrane. Bacteria are able to detect the local bacteria density. This information is used to coordinate processes that are only efficient if many of the population do them at the same time.

Another potential source of inspiration is "amorphous computing," a general concept was introduced by Abelson et al. [2], Bachrach and Beal [22]. The investigations reported under the name of amorphous computing are focused on finding programming paradigms for a continuous "smart" medium or rather for a discrete approximation of such a medium. It is composed of immobile particles and can be thought of as being similar to a sensor array (cf. sensor/actuator networks). The approach is biologically inspired by the cooperation of cells in natural multicellular organisms, the hardware is assumed to be unreliable, and the network topology is assumed to be unknown. Essentially, it is research on programming languages that are defined on the continuous medium abstraction and the challenge is to find compilers that translate these abstract descriptions into executable code for the sensor nodes [426–428].

1.5 Homogeneous and Heterogeneous Swarms

As discussed, the standard robot swarm is supposed to be homogeneous to allow for mass production, high redundancy, and robustness. However, there are also concepts of heterogeneous swarms [98, 105, 209, 318]. In a heterogeneous swarm, the complexity of an individual robot can be further decreased, even though in the

homogeneous approach a robot is already supposed to be simple. Following the heterogeneous approach, the robots can be designed as a small set of different robot types that can then be put together in appropriate sub-sets to enable the group to complete the respective task. Heterogeneity lowers the degree of redundancy, which may reduce robustness if not considered carefully.

As a side comment, notice that there are also concepts of intentionally introducing anti-collaboration as conflicts into swarms in the form of "anti-agents" or "anti-components" [272, 273]. For example, in an aggregation task one could introduce robots that do not aggregate but rather command clusters of aggregated robots to dissolve the cluster. Such an approach may be rather counterintuitive but can increase an algorithm's performance. Related work is that of two swarms operating independently but within a shared area [45].

1.6 The Human Factor

Although many applications of swarm robotics focus on ideas such as automation and operating in environments that are dangerous for human beings, the human factor should not be ignored. Robot swarms might operate in environments shared with human beings or individual robots within the swarm might be controlled by a human remotely. An example study of human–swarm interaction is reported by Kapellmann-Zafra et al. [207]. They investigate how a human operator can steer the robots based on their local view only. A situation that one would face in reality is that there is no system that fuses the sensor data from all robots and creates at least a semi-global view for the human operator.

Given that the public perception of swarm robotics is rather negative because of biased science-fiction stories, it is surprising that there are not many studies investigating the psychological aspect of robot swarms. A positive exception is the study by Podevijn et al. [311] who investigate the influence of robot swarms on the psychophysiological state of humans. They have found that "the number of robots to which a human is exposed has a significant impact on the psychophysiological state of the human and that higher numbers of robots provoke a stronger response." That may be the expected result and most of us can understand that having many agents (robots, animals, human beings) around ourselves and close to ourselves can be at least unpleasant depending on the context. Hence, we should investigate options of how robot swarms can be designed such that they are not perceived as mischiefs coming in a high number.

Another aspect of the human factor is the software engineer of robot swarm controllers and required or helpful interfaces between the human engineer and the robot swarms. McLurkin et al. [263] address that issue. Given that one works with possibly hundreds or thousands of robots, it is essential to allow for simple debugging that does not require to handle robots individually. The challenge is that, while an individual robot is already complex in itself, the emergent robot group dynamics may be even more complex. A structured, sophisticated design approach is hence even more important.

Finally, there is also the aspect of interfacing between a human supervisor and a robot swarm. Research on human–swarm interaction is discussed in Sect. 4.13.2. There is still a lot to do to optimize how human beings interact with robot swarms.

1.7 Implementations in Hardware and Software

Later in this book we go through a number of example scenarios of swarm robotics (see Chap. 4) and we also discuss hardware aspects of swarm robots in some detail (see Chap. 2). For now we only look quickly at some examples of scenarios and a few swarm robotics projects. Finally, we shortly discuss the use of robot simulators and potential future applications of swarm robotics.

1.7.1 Example Tasks and Swarm Robotic Projects

> In general it's probably better for them to live separately. They exist in loose swarms... However, they will unite in moments of danger
>
> —Stanislaw Lem, The Invincible

Many tasks that have been completed by robot swarms are based on motion of robots and detection of swarm mates. Examples for such tasks are aggregation, self-organization into a lattice formation or dispersion, flocking, mere covering of an area, and exploration tasks. Tasks of higher complexity include the application of additional, special-purpose sensors or necessitate to save acquired information, such as in the deployment of distributed antennas or arrays of mobile sensors, mapping of the environment, and creation of gradient fields (i.e., robots operate as guideposts). In manipulation tasks special actuators are needed, for example, in foraging behaviors, mining, and containment of oil spills. A particularly interesting kind of tasks include the interaction with a second group of agents, for example, shepherding, that is, a robot swarm herds a group of animals. In summary, the list of potential tasks is long and not necessarily different from the list of applications that can be solved by a single robot (see Chap. 4). However, a swarm of robots heavily parallelizes the execution of the task and adds further qualities, such as robustness and scalability.

There have been already many research projects with focus on swarm robotics during the last 15 years. Here, we shortly summarize the ideas and objectives of three of them. One of the first projects on swarm robotics was the project *Swarm-bots: Swarms of self-assembling artifacts*[4] [100]. It lasted three and a half years from 2001 until 2005. The main objective was to design and control robots that

[4] http://www.swarm-bots.org/, funded by the European commission, grant FET IST-2000-31010.

self-organize and self-assemble inspired by swarm intelligence and the behavior of social insects. The *Swarm-bots* were designed as small mobile robots with the special capability of being able to physically connect to each other. They have a small gripper directly attached to the main robot body that can grab a circumferential ring of a close-by robot. This way the robots can aggregate, physically connect, and still move in the assembled state. One can say that the assembled robots form a bigger robot consisting of autonomous robot modules. An advantage of the assembled, bigger robot is that it can pass steeper slopes and bigger holes that would be impassable for a single robot (so somehow it implements the vision of the above quote from Lem's *The Invincible*). Another notable showcase was that of 18 swarm-bots aggregating in four lines and pulling a little girl laying on the floor. Here, the idea is that robots can work together and apply bigger physical forces together.

Another interesting mention is the project *I-SWARM: Intelligent Small World Autonomous Robots for Micro-manipulation*[5] [358]. It lasted four years from 2004 until 2008. The very ambitious objective was to build 100 extremely small robots of dimensions $1\times1\times1$ mm^3. These extreme requirements were later released a bit to dimensions $3\times3\times3$ mm^3. The engineering challenge in *I-SWARM* was to integrate novel actuators for locomotion, miniaturized powering with micro solar cells, miniaturized wireless communication, and integrated circuits for an on-board intelligence. Also here the idea was to control the swarm of tiny robots by methods from swarm intelligence. In the end the project did not deliver fully functional robots due to limited resources but the developed technologies, the methodology, and the designs are ready to use.

The third project to mention here is the project *TERMES: Collective construction of 3D structures by termite-inspired climbing robots*[6] [306]. The project was active in the years between 2011 and 2014. The objective of *TERMES* was to develop a swarm construction system (collective construction) based on collaborating robots that pick up building material, transport it, and release it on the construction site to build desired structures. The approach is roughly inspired by the behavior of social termites that build amazingly complex mounds. We will discuss the *TERMES* project in more detail in Chap. 4.

1.7.2 Simulation Tools

Conducting a swarm robotics experiment is typically expensive because many robots need to be taken care of. Especially in experiments with 50, 100 [395], or even 1000 robots [332] even otherwise simple procedures, such as charging, flashing, or initial positioning of the robots, become tedious tasks. Also wear,

[5]Funded by the European commission, grant IST FET-open 507006.

[6]http://www.eecs.harvard.edu/ssr/projects/cons/termes.html, funded by the Wyss Institute for Biologically Inspired Engineering, Harvard.

reliability, and robustness of the hardware are issues of course. Hence, it is useful to do (at least initially) simulations. There are many robot simulators but not all of them are scalable themselves. Experiments with up to 1000 robots are challenging even software-wise in simulation.

A simulation framework that is dedicated to swarm robotic experiments is *ARGoS*[7] [308]. *ARGoS* has a flexible, modular design which is open for different robot platforms, parallelizable, and adjustable to different levels of abstraction. It was under development for almost ten years, is still maintained, and has been used for many different projects and is currently in use at many institutions in research and teaching.

Other simulators, that are also dedicated to swarm robotics, are the recently reported tool *Kilombo* [196] and the commercial *Webots* robot simulator.[8] *Kilombo* is specialized to simulate Kilobots, a robot design that is open-hardware but is also commercially distributed [331]. *Webots* is a general-purpose tool to simulate robots and is open for different robot platforms.

Beyond these special swarm robotics simulators, there are a number of open-source projects for robot simulations and multi-agent simulations. Many of them have been around for many years but also many of them are neither perfectly maintained nor well documented. Examples are breve [218], player/stage [142], and MASON [243]. Another option is to use game engines as a starting point for custom development of a simple robot simulator. For example, one can use *Box2D: A 2D Physics Engine for Games*[9] that provides collision detection techniques and allows to rapidly prototype sensors and robot platforms.

1.7.3 Future Applications

As many other research topics connected to mobile robotics, also swarm robotics has not yet many real-world applications that are already on the market. One of the few exceptions is the application of underwater exploration and monitoring which is in use in research applications [103, 275]. The breakthrough of mobile robotics applications are expected by many for the next years. This is not the first time in history that a major breakthrough is predicted for robotics [139], hence one should carefully observe what is going to happen.

Besides the often recited standard applications of cleaning, monitoring, exploration, and guarding, there is a vast variety of ongoing research projects in swarm robotics and many of them have high potential for future applications. A recent example is *Zooids: Building Blocks for Swarm User Interfaces* [217, 228]. The fascinating idea is to use small mobile robots on a desk as user interface. Also high

[7] http://www.argos-sim.info/.

[8] https://www.cyberbotics.com/.

[9] http://box2d.org/.

potential has the idea of self-organized construction with swarm robots [184, 332]. For example, the consultancy *Arup* mentioned swarm robotics in its *Future Of Rail 2050* report.[10] They see applications in infrastructure repair and large-scale construction.

An intriguing set of long-term visions for swarm robotics are the following challenges. In the future, we may face the end of using antibiotics as we know it due to antimicrobial resistances. We may face the end of pesticides also due to resistances. Our oceans are littered with millions of tons of plastic. The future of space flight may be the development of robust and scalable swarms of probes.

Nanorobotics [67, 189, 233] is based on the idea to build many micro- or nano-scale robots that operate possibly even within the human body. They could be used to monitor, to deliver drugs carefully targeted, and could even fight bacteria. Hence, they could be an option once bacterial resistance against antibiotics will be a major problem and we will run out of effective antibiotics. A similar concept but on bigger length-scales and hence more realistic would be to use swarm robots in agriculture. Similarly to bacterial resistance there are so-called super weeds that are created by large-scale applications of pesticides and that are resistant against most pesticides. The application of swarm robotics to agriculture would mean to fight pests mechanically again instead of chemically. So we would go back to previous, possibly safer methods; only this time it would not be humans picking the plants and bugs but many small robots.

Besides everyday cleaning tasks, such as facade cleaning, we also face the huge task of cleaning our oceans that are littered with huge loads of plastic. Similarly, we will need to clean the space around our planet because satellites are endangered by even tiny pieces of garbage. Both of these cleaning tasks most likely need to be done by autonomous systems and both need to be heavily parallelized. There seems to be no chance of doing it in a centralized way just because of the sheer magnitude of the task. Similar techniques may prove to be useful in space flight, in general, because the robustness and scalability of swarm robotics will make space projects much more reliable and less of a gamble.

Besides these promising future applications there are also unpleasant developments in weaponry. It is clear that a robust and scalable technology is also attractive for the military and ongoing work focuses on the automation of war. This obvious thought is also reflected in fiction, where swarms of robots appear typically as threats. Unfortunately, there are already many projects that investigate military applications of robot swarms. This is, however, the curse of technology that there is dual use of it. The Swiss author Friedrich Dürrenmatt has summarized this dilemma in his play *The Physicists*: "Nothing that has been thought can ever be taken back." One feels tempted to add also: what can be thought will be thought. Many researches have taken actions to create awareness and also to do research on the subject of *ethical robots*[11] and *machine ethics* [92]. I desperately hope that what

[10] http://www.arup.com/homepage_future_of_rail.

[11] https://arxiv.org/abs/1606.02583.

happened to chemistry in World War I, gas war, and to physics in World War II, nuclear war, won't happen to robotics. As researchers and engineers we have to take the responsibility not by trying to prevent progress but by trying to prevent evil applications. Swarm intelligence and its application swarm robotics have a high potential to be of good use for humankind. We should use the novel technology of swarm robotics to solve a few of our many problems instead of using it to create even new problems.

1.8 Further Reading

Summaries of swarm robotics research are the review paper by Brambilla et al. [54], a Scholarpedia article by Dorigo et al. [96], and two review papers by Bayindir and Şahin [33] and Bayindir [32].

1.9 Tasks

1.9.1 Task: Scaling of a Computer System

We create a simple model of a computer system. The following is a simple example from queuing theory. We assume that new jobs for the computer system come in with a constant rate of α and are added to a waiting list. We also assume that the process is without memory, that is, incoming jobs are statistically independent from each other. Therefore, we model the incoming jobs as a Poisson process. The probability that i jobs arrive within a given time interval Δt is given by

$$P(X = i) = \frac{e^{-\lambda}\lambda^i}{i!} \quad , \tag{1.7}$$

for $\lambda = \alpha\Delta t$. In the following we simply set $\Delta t = 1$.

a. Plot $P(X = i)$ for a reasonable interval of X and $\alpha \in \{0.01, 0.1, 0.5, 1\}$.
b. Implement a program that samples numbers of incoming jobs from $P(X = i)$.
c. Implement a model that iterates over these two phases: calculate and administrate new incoming jobs, then operate on the current job (one at a time). Generate a sample of 2000 time steps for $\alpha = 0.1$ and a processing duration of 4 steps per job. What is the average length of the waiting list?
d. Change your program such that you can average the waiting time list length over many samples (independent runs of your model over 2000 time steps each). Determine the average list length for rates $\alpha \in [0.005, 0.25]$ in steps of 0.005 based on 200 samples and plot it.
e. Do the same for a processing duration of only 2 steps per job with rates $\alpha \in [0.005, 0.5]$ in steps of 0.005. Compare the two plots.

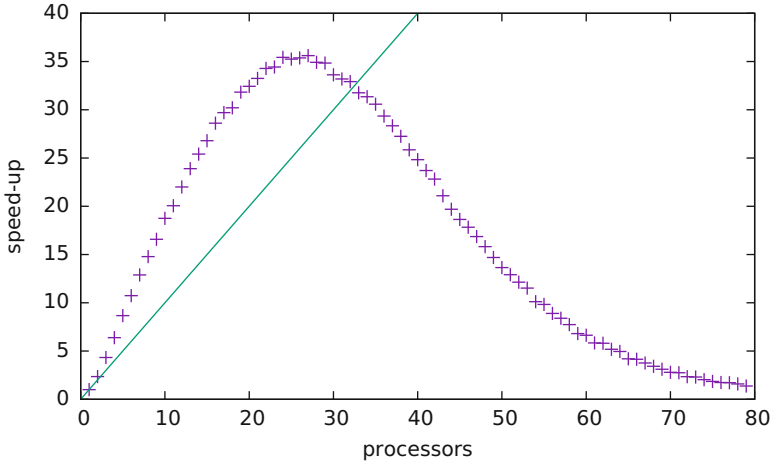


Fig. 1.12 Universal Scalability Law by Gunther [158]

1.9.2 Task: Superlinear Speedup

In the following we try to implement a simple simulation of a computer system that shows a superlinear speed-up as modeled by the Universal Scalability Law for $\alpha < 0, \beta > 0$ (see Fig. 1.12). To do so, we choose to simulate influences of cache coherence and cache hits in a simplistic way.

a. Implement a program that iterates over $t = 1000$ time steps. In each time step simulate with simple sequential code the parallel execution of p processors. Say the problem size is 100 and each processor gets a piece of it of size $s = \frac{100}{p}$. Each processor should do one of three different tasks in each time step.
Either it executes a task

   (a) to obtain cache coherency with probability

$$P_c(p) = \frac{1}{1 + \exp(-0.1(p - 30))} \quad , \tag{1.8}$$

   depending on the number of processors p;
   or if the cache is currently coherent $(1 - P_c(p))$ it completes
   (b) one job of the actual task (cache hit) with probability

$$P_h(s) = \frac{1}{2}\left(1 - \frac{1}{1 + \exp(-0.05(s - 15))}\right) + \frac{1}{4} \quad , \tag{1.9}$$

   for each processor's piece of the problem of size s;
   or it executes a task to

(c) load a page from memory (cache coherent but cache miss) with probability

$$1 - P_h(s). \tag{1.10}$$

For each execution of task 2 (completion of one job of the actual task) increment a variable that measures the performance.

b. Use your code to simulate different system sizes $p \in \{1, 2, \ldots, 80\}$. Adapt the size of each processor's piece of the problem $s = \frac{100}{p}$ accordingly. Keep track of the obtained performance (number of completed jobs) and calculate the speed-up for all settings (performance normalized by the performance for $p = 1$). Plot a diagram of speed-up over number of processors. Play around with probabilities $P_c(p)$ and $P_h(s)$. What is necessary to achieve a non-linear speed-up?

1.9.3 Task: Synchronization of a Swarm

Swarm systems are asynchronous systems. There is no central clock that could be accessed by everyone. If a swarm needs to act in synchrony it has to explicitly synchronize first. An example of a biological system that shows synchronization is a population of fireflies (see Fig. 1.13). "Though most species of firefly are not generally known to synchronize in groups, there are some (for example, *Pteroptyx cribellata*, *Luciola pupilla*, and *Pteroptyx malaccae*) that have been observed to do so in certain settings."[12] This is also related to the field of sensor networks and ad-hoc networks and their decentralized synchronization, see Tyrrell et al. [391].

In the following we create a simple model of such a firefly population. The population is scattered randomly over a square of 1 by 1 (uniform distribution). We assume that fireflies are stationary and can only perceive local neighbors in their vicinity. We say two fireflies are within the vicinity of each other if the distance between them is smaller than r. Hence, there is a virtual disc centered at each firefly's position and every other firefly sitting on that disc is a neighbor. The fireflies flash in cycles. We define the cycle length by $L = 50$ time steps. The firefly flashes for $L/2$ steps followed by $L/2$ steps of not flashing. This holds except for those cases when the firefly tries to correct its cycle to synchronize. In the time step after it has started to flash it checks its neighbors and tests whether the majority of them is actually already flashing. If so, the firefly corrects its clock by adding 1, that is, it is decreasing the current flashing cycle from $L/2$ to $L/2 - 1$ steps and will consequently flash 1 step earlier next time.

a. Implement the model for swarm size $N = 150$ and cycle length $L = 50$. Calculate the average number of neighbors per firefly for vicinity distances $r \in$

[12] http://ccl.northwestern.edu/netlogo/models/Fireflies.

Fig. 1.13 Synchronized flash of fireflies (Photo by xenmate, CC BY 2.0)

$\{0.05, 0.1, 0.5, 1.4\}$. Plot the number of currently flashing fireflies over time for vicinity distances $r \in \{0.05, 0.1, 0.5, 1.4\}$ for 5000 time steps each. When plotting the number of currently flashing flies, make sure you plot the full interval of $[0, 150]$ for the vertical axis.

b. Extend your model to determine the minimum and maximum number of concurrently flashing fireflies during the very last cycle (last $L = 50$ time steps starting from $t = 4950$). By subtracting the minimum from the maximum you get double of the amplitude of the flash cycle. Average the measured amplitudes over 50 samples each (50 independent simulation runs with 5000 time steps each) and plot them over vicinities $r \in [0.025, 1.4]$ in steps of 0.025. What seems a good choice for the vicinity and the swarm density?

Chapter 2
Short Introduction to Robotics

"I can't define a robot, but I know one when I see one."

—Joseph Engelberger

"A robot must know it is a robot"

—Nikola Kesarovski, The Fifth Law of Robotics

Abstract This is a little crash course in robotics for the case you haven't heard so much about robotics yet.

This short introduction to mobile robotics is starting from a general perspective and quickly introduces fundamental concepts, such as sensors, actuators, and kinematics. We continue with a short introduction to open-loop and closed-loop control. Agent models, behavior-based robotics, and potential field control are introduced as control options particularly interesting for swarm robots. We conclude with a presentation of several hardware platforms dedicated to swarm robotics, such as the s-bot, the I-SWARM robot, Alice, and the Kilobot among others.

Before we start with the actual subject of robot swarms, we have to go through some basic concepts of mobile robotics. We start with discussing what a single robot needs and at the end of the chapter we have a look at some examples of robots specifically designed for swarm robotics.

Similarly to other complex issues, such as intelligence or life, also the concept of a robot is not easily defined. One can say that a robot is a computer-controlled, autonomous machine built for real-world functions (see Fig. 2.1 for examples). However, it is not clear whether that definition is sufficient. The *Robot Institute of America* defines a robot as "a reprogrammable, multifunction manipulator" (1979)[1] and the *European Common Market* defines a robot as "an independently acting and self-controlling machine." When we define a robot based on what it needs to be complete, we can say that a robot needs manipulators to change either its

[1]New Guide to Science by Isaac Asimov, 1984.

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2_2

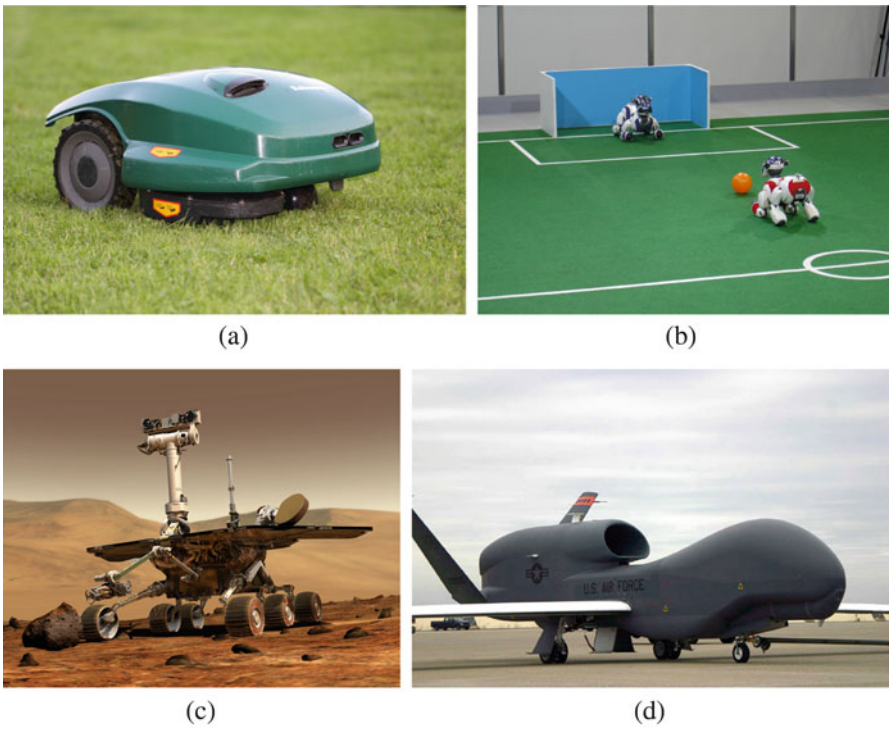


Fig. 2.1 Example robots. (**a**) Robomow® RM 400 (photo by Holger Casselmann, CC BY-SA 3.0). (**b**) Sony® Aibos at Robocup 2005 (photo by Alex North, CC BY-SA 2.0). (**c**) NASA Mars Exploration Rover (photo by NASA, public domain). (**d**) RQ-4 Global Hawk (public domain)

own position within the environment or to change its environment. A robot also needs sensors to (partially) perceive the current state of the environment, it needs a computer to process these inputs, and it needs to calculate actuator values that control the robot's motors. Usually, one adds that the robot should be a multi-purpose machine, that is, it can be applied not only for exactly one task but many different tasks.

We distinguish three classes of robots: mere manipulators, mobile robots, and hybrid robots. The manipulators are basically robot arms. The typical industrial robot is a robot arm and commonly found in many industrial settings. These kind of robots are very popular and have been around for many decades. However, for swarm robotics they have little or no relevance. In fact, they are rather boring machines, often even without sensors.

Mobile robots are in our main focus. They are unmanned vehicles that are capable of locomotion. An example are vacuum cleaning robots. Once a robot is mobile it definitely needs sensors to make sure it is not endangering others or itself. Proximity sensors, that measure distances to closeby objects, are useful here. Almost all robots used in swarm robotics are mobile robots.

The third class are hybrid robots. These are mobile robots that have manipulators. The stars of hybrid robots are humanoid robots, that is, robots with arms and legs similar to human beings. Their drawback is that they are rather complex machines and hence they are expensive. Hybrid robots are not of much importance in swarm robotics yet.

2.1 Components

Next we go through the different components of robots. Starting from a body (some basic static structure), a robot has effectors, actuators, sensors, a controller, and software. The layman may forget about the existence of software because it can't be seen but, as you know, it is the essential part of a robot. Other components are joints and additional structural parts.

2.1.1 Body and Joints

The body of a robot can be understood as a graph of links and joints. The links are parts that provide structural, physical properties. Joints connect these parts by constraining the spatial relation of two or more links. Once two links are connected by a joint, these two links cannot move fully independently anymore. What kind of movements are still possible without causing damage depends on the type of the joint. There are many different types and we just mention three here. There is the ball joint that allows rotation in three dimensions. Remember that our world is three-dimensional and positions are determined by three axes. An object can be moved up and down, left and right, forward and backward. In addition, objects can be oriented in different ways by rotating them around the three axes. From your flight simulation game you may know the three ways of rotation: yawning, pitching, and rolling. So naturally there are six so-called degrees of freedom and a joint restricts this set of possibilities. So the ball joint allows rotation in any of the three axes. An example in your body is the shoulder joint. A hinge joint allows only for rotation around one axis. Examples in your body are the joints in your fingers and toes, while the knee allows actually some rotation (however not much as many footballers learn the hard way). A slider joint allows a translation along one axis. For some reason the slider joint is not popular in nature. We don't have a slider joint in our body although it may be handy in certain situations.

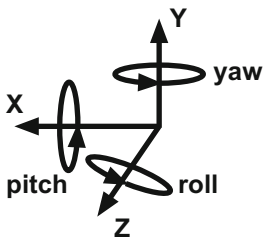


Fig. 2.2 Six degrees of freedom (DOF)

2.1.2 Degrees of Freedom

When you want to talk to a roboticist, make sure you have understood the concept of degrees of freedom (DOF). It is a popular approach to classify robots (especially manipulators) according to the total number of DOFs that they allow. For that approach the number of DOF is summed over all joints. As a rule of thumb (pun intended) you can use a three-finger rule. Your thumb, first, and second finger point in the three directions of possible translations (see Fig. 2.2). In addition, imagine the three rotations around each of them. How many DOF can you identify in your arm? A healthy human arm is considered to have seven DOF: shoulder pitch, shoulder roll, arm yaw, elbow pitch, wrist pitch, wrist yaw, and wrist roll.

2.1.3 Effector

A robot's effector (also end-effector) is the component that is used to accomplish some desired physical function. The effector in a complex robot can be a hand to grip objects, but more common would be a torch, wheels, or legs. Hence, the effector implements the manipulation task that we determined in the definition of robots. An effector, even if it is just a wheel, helps a robot to manipulate the environment or to reposition itself within the environment.

2.1.4 Actuator

The actuators can be considered as the robot's muscles. In the standard case of a mobile robot they are electric motors because they are cheap and easy to control. Many industrial robots, however, are driven by pneumatic or hydraulic systems. More fancy actuators can be piezoelectric motors or any other device that allows the robot to apply a physical force on command.

The most common actuation system in mobile robotics is the differential drive. Many robots just have two wheels but both of them are controlled by a distinct motor. By turning the motors at different speeds the robot can turn or even spin on the spot.

2.1.5 Sensor

Maybe the most interesting type of component is the sensor. They are essential to make a robot different from a computer as sensors allow the robot to perceive its environment. A robot doesn't have to wait for explicit user input like a computer, instead it can observe its environment and react autonomously to changes. Sensors come in two categories: active and passive sensors. An active sensor derives information from a reaction of the environment to a previous action of the robot. Examples are bumpers, infrared, and sonar sensors. The bumper detects an object by having the environment pushing it into the robot. It is considered active because here we think of mobile robots that actively move. However, one could argue that a moving object could push the bumper even if the robot is static. The case is clearer for infrared and sonar sensors. They functionally rely on the concept of first sending out a signal, light or sound respectively, and then waiting for the echo. By analyzing this received signal the robot can determine certain properties of the environment. In most cases infrared and sonar sensors are used to measure distances to closeby objects.

Infrared and sonar sensors belong to the class of range finders. These are sensors that are used to determine distances. Other examples are radar, lasers, whiskers, and even GPS. The problem with active sensors used as range finders is interference. Once you have too many sensors operating in a small area they interfere with each other. Then a sensor cannot distinguish anymore between echos of its own signal and signals from other sensors. One could have too many sensors in one place because there are many on one robot or, typical for swarm robotics, too many robots with active sensors in one place. Interference is a challenging problem and there is no simple solution. However, one option is to limit the range of the used sensors, hence scaling the problem down. The range of an active sensor can be restricted by limiting the transmission power (smaller amplitudes in the light/sound signal).

Imaging sensors are typical examples for passive sensors. They create a visual representation of the robot's environment. Besides usual cameras, also stereo vision systems are used to add depth information. Using vision is not very common in swarm robotics because it typically requires a lot of computations. Often swarm robots, however, have not very powerful microprocessors and the computational load roughly increases with each additional pixel. In some cases, swarm robots have low-resolution cameras with a low number of pixels or even only a single line of pixels.

Another class of sensors are proprioceptive sensors. These are sensors that gather information about the robot's internal state, for example, positions of joints. Proprioceptive sensors look inside the robot instead of monitoring the environment. A typical example is a shaft decoder (see Fig. 2.3) that counts the revolutions of an axis. It is built based on a light barrier and a wheel with holes on the axis that sometimes blocks the light barrier and sometimes not, depending on its angular position. By counting the alternating phases of light and no light the position of the axis and how often it turned can be detected. This gives useful information for configuring the robot and to be used in the robot's odometry.

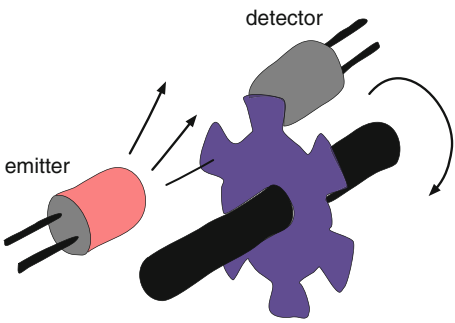


Fig. 2.3 A shaft decoder as example of a proprioceptive sensors

2.2 Odometry

Odometry is an important field within robotics that works on a problem, that we as human beings cannot fully connect to. Odometry is the estimation of distance and direction from a previous position of the robot. The underlying idea is that the robot always knows where it is. Usually, whenever I would ask you where you are, you could give me a quite clear answer, even specifying the estimated distance to the next door and the room's walls if necessary. For a robot that is not a simple task. Remember, a robot may not have vision and only a few range finders. If you like, a robot is somehow locked into a tin can and knows only little about its environment. If you had to close your eyes while wearing a noise-canceling headset, not being allowed to move your arms, and only being told distances to the next object to your front and back, navigation would also become tricky for you.

You can view odometry as a form of dead reckoning, that is the art of determining your position based on a known initial position and subsequent headings and speeds. Probably you remember stereotypical scenes from some submarine movie, where the captain is doing exactly that on a nautical chart (yet another example of people locked in a tin can). You may immediately guess that uncertainty is an issue in odometry. Errors in determining the robot's correct speed and heading will sum up and the determined position may be different from the robot's actual position. Even if the robot has perfect information about its initial position and measures the revolutions of its axes with proprioceptive sensors, errors will occur, sum up, and the robot will determine an incorrect position after a while. Please note that GPS is not always the simple answer here because we may need higher accuracy to safely navigate or we may operate indoors.

2.2.1 Non-systematic Errors, Systematic Errors, and Calibration

In odometry there are two classes of errors depending on their source. There are non-systematic errors that can usually not be measured and hence cannot be included in a model. These errors can be caused by uneven friction of the surface, wheel slippage, bumps, and uneven floors. Systematic errors are caused by differences between the model, that the robot has of itself, and the actual behavior of the robot. These systematic errors can be measured and then be incorporated in the model. We call these corrections of the model calibration. Another unexpectedly difficult task in robotics is to make robots go straight. Especially in swarm robotics, where we use small and simple components and the robots are lightweight, making robots go straight is nontrivial. Calibration can help here to compensate for nonsymmetrical features in the hardware. Typical examples are unequal wheel diameters, uncertainty about the wheel base (distance between axes), or just faulty axle bearings.

The uncertainty about the robot's position grows over time due to systematic and non-systematic errors. In classical robotics standard techniques such as Bayesian filters, particle filters, and Kalman filters are used. Here we don't go into these details because they are out of this book's scope. Instead we have a quick look into map making and then make a little excursion into how animals (here: ants) solve the problem of localization and navigation.

2.2.2 The Art of Map Making

The one moment when we as human beings also have trouble in locating ourselves is when we have to navigate a new city. For that purpose we have navigation systems and smart phones with map apps. In former times we would have used an actual map on paper. Hundreds of years ago map making was a true art because a lot of information about regions, countries, or whole continents was just missing (see Fig. 2.4). Robots face similar challenges whenever they are deployed in a new area. Instead of relying on odometry exclusively they can also build a map on the fly. This is another popular field within robotics, called simultaneous localization and mapping or short SLAM. There are many different ways of representing the environment in the form of a map. Examples are maps based on a grid (filling grid cells where objects and walls were detected), landmarks (e.g., features that are recognizable by vision), or a laser scan (i.e., processing many detected points in 3-d).

Fig. 2.4 The art of map making: sea map of Portugal by Lucas Janszoon Waghenaer (1584)

2.2.3 Excursion: Homing in Ants

In swarm robotics we often take inspiration from biological systems. We do that not as a purpose of its own but to get novel design ideas of how we can engineer robots. Swarm robots need to be simple and cheap and relatively simple animals, such as insects, can be a true inspiration. So how do ants solve the problem of localization and navigation? They face the same problem as they need to navigate, for example, from a feeding place back home to their nest to transport food to the colony. Wolf [423] considers three methods of how ants can do that (see Fig. 2.5). In the case of route following, an ant can drop pheromones on its way to the food and then follow them back to the nest. In the case of path integration, an ant can remember the general heading and the number of steps towards the nest with help, for example, from the terrestrial magnetic field. In the case of map-like navigation, an ant can remember several landmarks on the way and navigate back by approaching each of them sequentially. For each of these methods at least one ant species does exist that navigates that way.

If you are wondering how an ant could possibly count its steps, then you can of course take the skeptic position and doubt it. However, biologists have done an experiment to test this hypothesis. The experiment is rather cruel but in most countries (e.g., within the European Union) there is no legislation protecting insects in the context of scientific experiments. Wittlinger et al. [422] have changed the step length of a group of desert ants (*Cataglyphis*, that is the ant in question) by cutting off parts of their legs (i.e., shortening their step length) and extending the

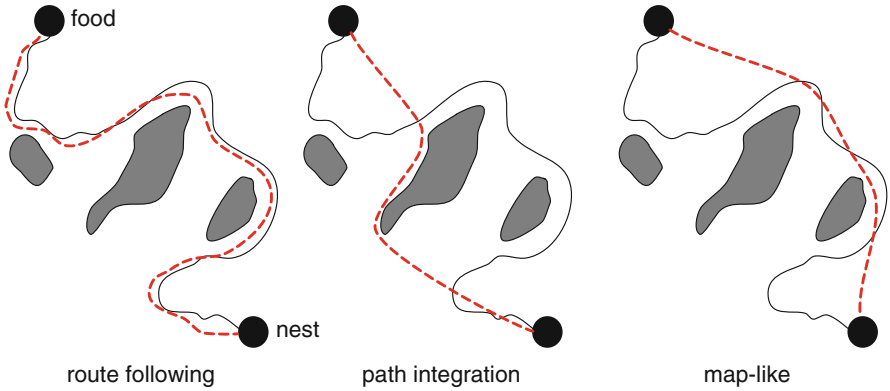


Fig. 2.5 Trajectories of homing ants from a food place to the nest for three methods: route following (e.g., by pheromones), path integration (e.g., by the terrestrial magnetic field and the number of steps), and map-like navigation (e.g., by remembering landmarks) [423]

legs of another group of ants (i.e., extending their step length). When these ants were left to navigate back to their nest, indeed those with the longer step length passed the nest and those with the shorter step length stopped short. Hence, this proves that the desert ant navigates by path integration based on step counting. It seems reasonable that they cannot actually count their steps but they can obviously estimate the number well enough.

If these methods were found by natural selection after many, many generations of ants, then they are probably also useful to be applied in swarm robots. Only, we would count revolutions of axes instead of counting steps as long as we usually don't have legged swarm robots.

2.3 Kinematics

The calculations of a robot's position by using odometry are an example of kinematics. Kinematics is the study of motion without regarding the causing forces. It ignores concepts, such as torque, force, mass, energy, and inertia. Hence, it is a kind of simplified physics only paying attention to the time-based and geometric constraints. Kinematics is yet another popular field within robotics because kinematics is especially relevant for robot arms and industrial robots. There are two kinds of kinematics: forward kinematics and inverse kinematics.

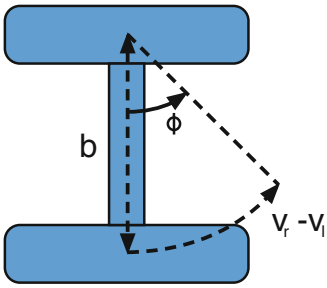


Fig. 2.6 Kinematics of the differential drive

2.3.1 Forward Kinematics

The problem of forward kinematics is stated in the following way. I give you the starting configuration of the robot arm including the angles of all its joints and some desired changes of some joints. Your task is then to calculate the new configuration. Note, this is a typical setting for industrial robots. So you would need to calculate the position and orientation of the end effector and all the joints.

A typical case for a swarm robot is the kinematics of differential steering (i.e., two wheels, each driven individually, see Fig. 2.6). The robot system is described by three variables: position of the robot's center of mass (x, y) and the robot's heading ϕ. In a problem of forward kinematics, I would give you the current robot position (x_0, y_0), its heading ϕ_0, and a desired change in terms of wheel speeds v_r for the right wheel and v_l for the left wheel. Given also the robot's track b (distance between wheels that share an axis), you can calculate the robot's speed m and write down the appropriate differential equations

$$dx/dt = m(t)\cos(\phi(t)), \tag{2.1}$$

$$dx/dt = ((v_r + v_l)/2)\cos(\phi(t)), \tag{2.2}$$

$$dy/dt = m(t)\sin(\phi(t)), \tag{2.3}$$

$$dy/dt = ((v_r + v_l)/2)\sin(\phi(t)), \tag{2.4}$$

$$d\phi/dt = (v_r - v_l)/b. \tag{2.5}$$

dx/dt is the first component of speed (in 2-d) and dy/dt is the second component of speed. $d\phi/dt$ is the arc change over the radius (angular speed). In order to get nice functions of time we integrate them all and get

$$\phi(t) = (v_r - v_l)t/b + \phi_0, \tag{2.6}$$

$$x(t) = x_0 + \frac{b(v_r + v_l)}{2(v_r - v_l)}(\sin((v_r - v_l)t/b + \phi_0) - \sin(\phi_0)), \tag{2.7}$$

$$y(t) = y_0 - \frac{b(v_r + v_l)}{2(v_r - v_l)}(\cos((v_r - v_l)t/b + \phi_0) - \cos(\phi_0)). \tag{2.8}$$

The turn radius for a circular trajectory is given by $\frac{b}{2}\frac{v_r+v_l}{v_r-v_l}$.

Fig. 2.7 The so-called "Canadarm2" robot arm of the International Space Station (photo by NASA, public domain)

We should not blindly trust these mathematical formulas but check for which cases we may have issues. There is the term $v_r - v_l$ in the denominator, that means, we have an asymptote (singularity) for $v_r - v_l \to 0$. For this case special handling is required. Fortunately, $v_r - v_l \to 0$ means that the robot is going straight and we could use another, simpler model.

2.3.2 Inverse Kinematics

The inverse kinematics problem is to calculate the appropriate parameters for all joints for a given position of the end-effector. For example, this could arise in situations as shown in Fig. 2.7. It is important to avoid collisions, not only of the robot arm with other structures but also of the robot arm with itself. This is mathematically a difficult problem because there are incorrect configurations, singularities, and also multiple possible solutions for certain problems. Also inverse kinematics is an intensively worked on field in robotics.

2.4 Control

Finally, we have to talk about robot controllers. This is probably the most exciting component of a robot because it is basically the robot's brain. We all want intelligent robots but obviously it is a challenge to program robots to behave intelligently in almost all situations. The controller directs the robot how to move and act. In

classical robotics there are two main controller paradigms: open-loop control and closed-loop control. An open-loop controller executes the robot's movement without paying attention to any feedback, that is, there is no chance to make corrections. The closed-loop controller executes the robot's movement and checks the progress with sensors. Hence, with closed-loop control we can compensate for errors.

If we want to control a mobile robot to make it drive parallel to a wall with an open-loop approach, then it is likely to veer off the path. The movement itself can be noisy, there could be slippage, model inaccuracies, or bumps. Open-loop control can hence not be applied in most cases working with mobile robots. For industrial robots open-loop control is more often an option. With a closed-loop control we can make use of a proximity sensor. We could position the sensor at the robot's side facing the wall. If the robot veers away or toward the wall, we can detect that and compensate for it.

2.4.1 Trajectory Error Compensation

When trying to keep a robot (or its end-effector) on a certain path, it will usually veer off eventually. Hence, we need a controller that compensates for that error. You can also think of the cruise control of a car and how you would program a controller to change the car's acceleration appropriately. There is a classical control approach called PID control (see Fig. 2.8). P, I, and D are components of a controller that can be combined for best results and they are distinguished by their mathematical property. The P-component provides a force in negative proportion to the measured error. The D-component adds a force that is proportional to the first derivative of the measured error. Hence, it measures the speed of change of the error. For example, an error could get quickly bigger or not change at all. The I-component adds a force that is proportional to the integral of the measured error. Hence, it sums over the past errors and is big if the error was big for quite a while in the past. Combining all three components gives you a PID controller that is well prepared for many different dynamics in the error. Depending on certain parameters you can configure a PID controller (see Fig. 2.9) to ensure it is neither too aggressive (oscillations) nor too slow.

Unfortunately, we cannot easily apply this concept in mobile robotics, at least not in the actual challenging situations. In most cases we would not even know the exact desired actuation value, which means also we could not calculate an error. There are more complex challenges in mobile robotics, such as deciding about when to explore the environment and when to exploit what is already known. Should your robot try to pass the next door to check another room or should it stay in the current room? These questions cannot be resolved with a PID controller. Hence, it is obvious that we need additional approaches.

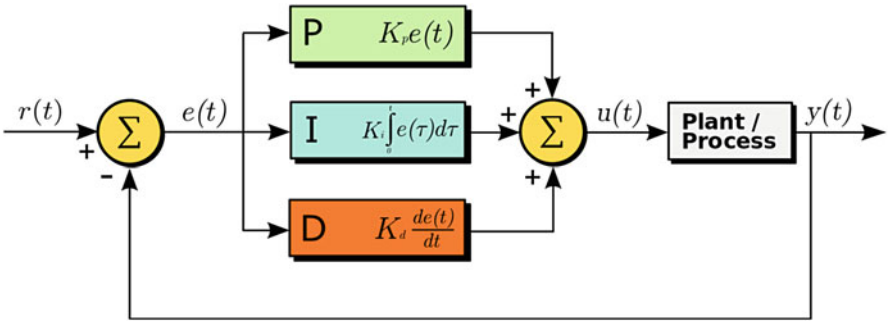


Fig. 2.8 Proportional-integral-derivative controller, short PID controller (schema by Arturo Urquizo, CC BY-SA 3.0)

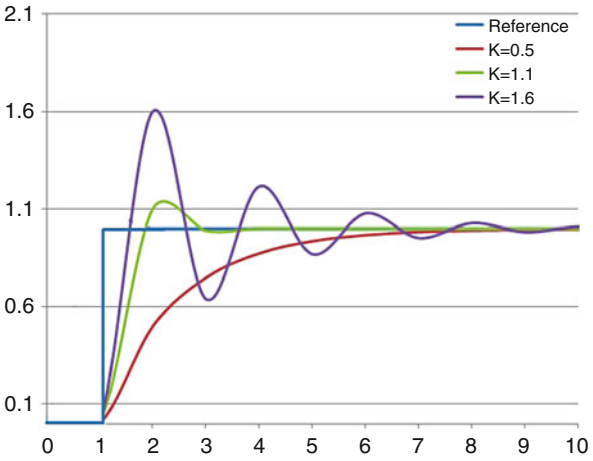


Fig. 2.9 Evolution of the error over time for different parameters. The horizontal axes gives time, the vertical axis the robot's position. The desired position suddenly changes at $t = 1$. A too aggressive controller overshoots and oscillates and a too slow controller compensates without overshooting but takes too long (diagram by TimmmyK, CC0 1.0)

2.4.2 Controllers for Swarm Robots

Not only the hardware of swarm robots is supposed to be kept simple but also their software, that is, their control. The idea is to rely instead on processes of self-organization as mentioned before. By collaboration the robots are still able to solve complex tasks that an individual would not be able to solve with its simple controller. The standard control approach in swarm robotics is hence based on reactive control. Russell and Norvig [337] introduce classes of agent models. The simplest model is that of the simple reflex agent. A simple reflex agent has only simple condition-action rules (if-then rules) that determine how it behaves (e.g., "if the sensor value of proximity sensor 5 is below 33, then make a left turn"). This is a purely reactive behavior because the robot just directly reacts to its current perception. A history of perceptions is not considered. The second agent model in the hierarchy of Russell and Norvig [337] is the model-based reflex agent. This agent has already an internal state that allows the robot to potentially react differently to the very

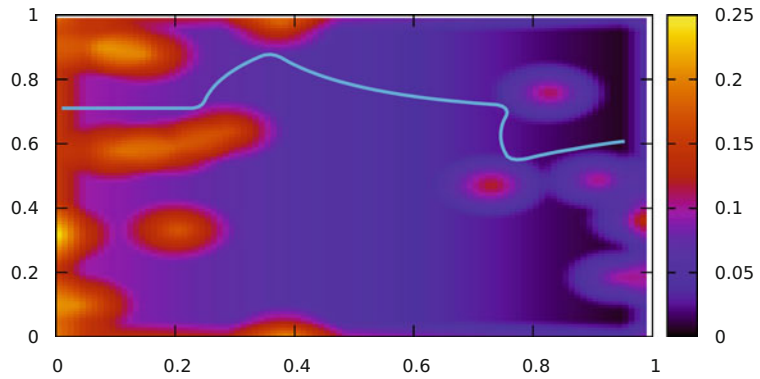


Fig. 2.10 Example trajectory of a robot in 2-d space controlled by a potential field (colors encode height of the potential field)

same perception. The behavior of the robot is still determined by condition-action rules while considering a history of perceptions. The internal state of a model-based reflex agent can be simple as in a finite state automaton or it can be more complex in the form, for example, of an array of floating point values used to store previous perceptions.

The concept of equipping robots with reactive controllers only is related to the so-called behavior-based robotics by Arkin [13] and the concept of subsumption architecture by Brooks [56, 57]. Also from the point of view of biology considering, for example social insects, reactive behavior seems plausible [48].

The idea of reactive control is easily understood by the example of so-called potential field control. Using potential field control, the problem of collision-free navigation is translated into a virtual physical world. The potential field is constructed based on the robot's perception and based on the defined task. As an example, the robot in Fig. 2.10 is supposed to navigate from the left to the right. This is implemented by an overall slope in the potential field with the lower end on the right-hand side. Obstacles are represented by cones. The navigation of the robot can be imagined similar to a marble rolling down the slope while being repelled from the cones. This is a reactive behavior because the robot does not plan its actions ("Where will I end up when I do this now?") and the robot's actions are directly determined by the currently present slope of the potential field. However, depending on the implementation, past actions may influence the robot via its current acceleration that may integrate over past influences.

In swarm robotics, there is a lot of research using methods of machine learning [337], such as evolutionary swarm robotics [100, 383]. This is the application of evolutionary algorithms [109] to swarm robotics. Usually the weights of artificial neural networks are optimized using evolutionary computation to teach a swarm to solve a given task. However, there is also a lot of research where the controllers, for example, in the form of finite state machines, are simply hand-coded. The

complexity of both the swarm behavior itself and the challenge of finding the right controller have their origin in the self-organized interactions between robots and between robots and the environment. Even though the controllers themselves are simple, finding them is difficult. We will discuss that in depth later (see Chap. 5).

2.5 Swarm Robot Hardware

Finally, we have a closer look into a few selected examples of swarm robot hardware. The most common design is a rather small mobile robot (say, diameters between 4 and 12 cm) with differential drive and infrared sensors for proximity sensing, however, there are also some special designs.

2.5.1 s-bot

One of the most iconic hardware designs is the "s-bot" (see Fig. 2.11) of the swarm-bots project [288]. By today it is a bit outdated but back in 2004 it was an amazing approach to swarm robotics and robotics. It is a considerably small robot of 12 cm diameter, 15 cm height, and a weight of 660 g. Its two LiIon batteries allowed energy autonomy for more than 1 h. The robot's controller is based on a 400 MHz custom XScale CPU board with 64 MB of RAM and 32 MB of flash memory. The robot was accessed by WiFi, which is a great feature for debugging. The robot's actuators are two treels (i.e., a combination of wheels and tracks), turret rotation, rigid gripper elevation, and a rigid gripper. The original design included a 3-DOF side arm and a side arm gripper, which were later left out. The robot has eight RGB light-emitting diodes around the turret and red light-emitting diodes in grippers. The robot has an amazing number of sensors: 15 infrared sensors around the turret, four infrared sensors below the robot, position sensors on all degrees of freedom except the gripper, force and speed sensors on all major degrees of freedom, two humidity sensors, two temperature sensors, eight ambient light sensors around the turret, four accelerometers, which allow three-dimensional orientation, a 640×480 camera sensor with a custom optic based on a spherical mirror that provides omnidirectional vision, four microphones, two axis structure deformation sensors, and an optical barrier in grippers. Many experiments were reported using this robot, including experiments in evolutionary swarm robotics [100, 383]. The basic design of the s-bot was later redesigned in different versions called "marXbot" [49] and foot-bot within the Swarmanoid project [98].

Fig. 2.11 Five s-bots (12 cm diameter) showing their ability of physically connecting to each other [288] (photo by Francesco Mondada and Michael Bonani)

2.5.2 I-SWARM

Another big step forward in swarm robotics hardware research was the I-SWARM project [358] as described in Sect. 1.7.1. The extremely ambitious objective of I-SWARM was to develop a tiny swarm robot of size $1 \times 1 \times 1$ mm^3. Although this constraint was later relaxed a bit to $3 \times 3 \times 3$ mm^3, it is obvious that such a design comes with many challenges. The robot (see Fig. 2.12) has no battery but is powered by a highly efficient solar cell. The robot has no wheels, instead it is locomoted by three vibrating piezoelectric legs (slip-stick locomotion). Robot–robot communication is implemented by four infrared transceivers. The robot has a vibrating piezoelectric needle as sensor and the onboard electronics is a re-programmable ASIC (i.e., custom chip design). The final robot was not fully functional due to issues in the ASIC because there was not enough funding to do several design iterations. Still, it was a great achievement and the knowledge is there now. Any company with enough funding could start building this tiny robot. Potential applications could be cleaning and monitoring tasks in hardly reachable parts of machines.

2.5.3 Alice

The microrobot "Alice" is another example of a rather small swarm robot (see Fig. 2.13). It was developed by Caprari et al. [62] and has a size of $22 \times 21 \times 20$ mm^3 or 9.24 cm^3. It has a PIC microcontroller, a NiMH rechargeable battery, and flexible PCBs. A true exception were the two "Swatch" motors provided by the watchmaker Swatch. At that time small electro-motors were rare.

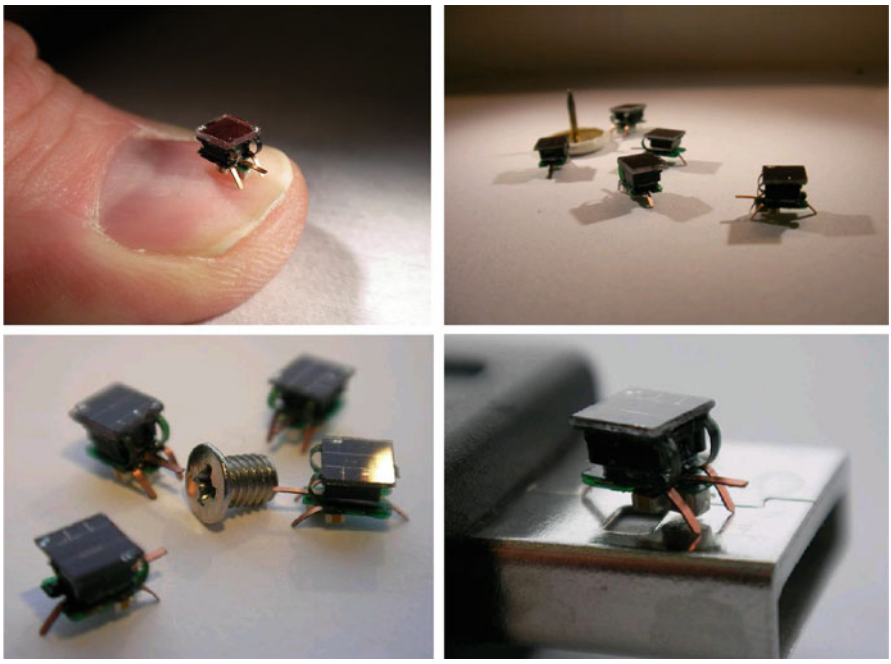


Fig. 2.12 The tiny swarm robot developed by Seyfried et al. [358] in the I-SWARM project (size $3 \times 3 \times 3$ mm^3)

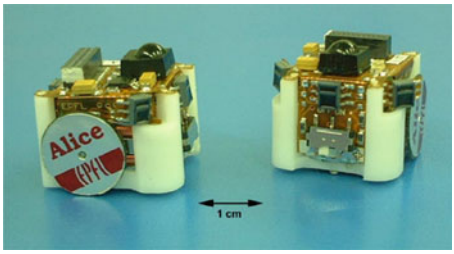


Fig. 2.13 The Alice microrobot, size of $22 \times 21 \times 20$ mm^3 [62] (photo by Gilles Caprari)

2.5.4 Kilobot

A swarm robot that is currently popular in swarm robotics research is the "Kilobot" [331, 376]. It is also relatively small with a diameter of 33 mm and a height of 34 mm (see Fig. 2.14). The overall design of the Kilobot seems to be inspired by the I-SWARM robot, because it also has vibrating legs instead of wheels (slip-stick locomotion). The controller is an Atmega 328 (8bit, 8MHz), it has 32 KB flash memory, 1 KB EEPROM, and 2 KB SRAM. The robot has a Li-Ion battery. Communication of up to 7 cm is done via infrared light that is reflecting off the ground surface (the infrared transceiver points downwards). Distance information of neighboring Kilobots can be estimated via signal strength. The robot has an RGB

Fig. 2.14 The popular Kilobot (diameter of 33 mm) [331]

LED, an ambient light sensor, and two vibration motors. They are independently controllable, hence, implementing a differential drive. This robot was used in the biggest robot experiment reported so far [332]. The experiment was done with 1024 robots. Hence, the name Kilobot was seemingly given in anticipation of that experiment.

One disadvantage of the Kilobot is that it is very slow due to the vibrating leg technique. Another issue not directly related to the robot is the difficulty of handling such large numbers of robots in experiments. All robots need to work reliably but, for example, for recharging they need to be handled. If a human operator has to manually handle them and maybe even calibrate them, then handling 1000 robots comes with a lot of overhead. Even if each of the 1000 robots only needs attention for two minutes, it will take more than 33 h to finish this task.

2.5.5 Other Swarm Robots

There are even more swarm robots. A class of swarm robots was designed explicitly for education, which doesn't mean they are not used in research as well. Examples are the "Thymio II" robot [328] and the "e-puck" robot [285]. Both are small mobile robots with differential drive and infrared sensors for proximity sensing and communication. Especially Thymio-II is used to attract children to robotics, which is a great service to both the public and the robotics community. A recent robot

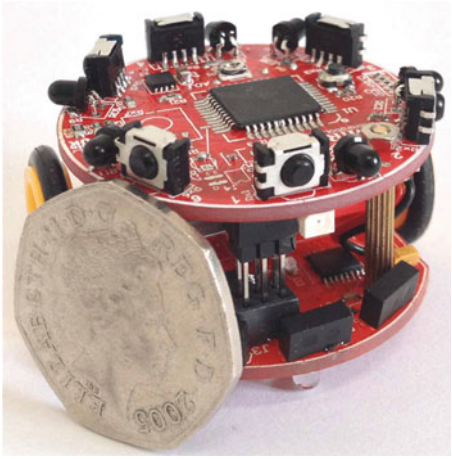


Fig. 2.15 The open platform robot Colias by Arvin et al. [16] (diameter of 4 cm) (photo by Farshad Arvin)

design for education is the "Pheeno" robot [416]. It has a modular design and focus seems to be on a more classical robotics approach including vision and gripping.

The robot "Colias" (see Fig. 2.15) was designed by Arvin et al. [16] as a low-cost, open platform with a diameter of 4 cm, a relatively high maximum speed of 35 cm/s, and the typical infrared sensors.

Other robot platforms following the standard differential-drive concept are "r-one" [261], "Wanda" [214], and "Jasmine" [197]. One of the earliest robot designs in swarm robotics was the "Khepera" robot by Mondada et al. [286], which is barely used anymore. The recently reported GRITSBot by Pickem et al. [307] was designed as an inexpensive swarm robot "to lower the entrance barrier." A robot that is unique because it neither relies on wheels nor vibrating legs is the robot reported by Elmenreich et al. [113] (see Fig. 2.16). It is a six-legged robot with hexapod locomotion based on a platform of a remote-controlled toy. The Droplet robot by Farrow et al. [116] is another robot based on slip-stick locomotion. It has omnidirectional motion based on six linear directions and it can turn in place.

Other special platforms relevant to swarm robotics are some approaches from modular robotics, where autonomous robot modules can physically attach to each other but also operate as independent entity, as done in the projects SYMBRION and REPLICATOR [240]. There are also a few projects doing underwater swarm robotics, that is, they develop autonomous underwater vehicles (AUV). Examples are the robots developed in the project CoCoRo [350] and the AUV called MONSUN (see Fig. 2.17) [300]. Especially communication between robots is challenging underwater. Radio cannot be used because electromagnetic waves do not propagate far in water. While CoCoRo uses visible light for communication, MONSUN uses an underwater acoustic modem, which has a much bigger range than light. However, also dealing with acoustic waves underwater is challenging because they are reflected at the surface, possibly at confining walls, or even at

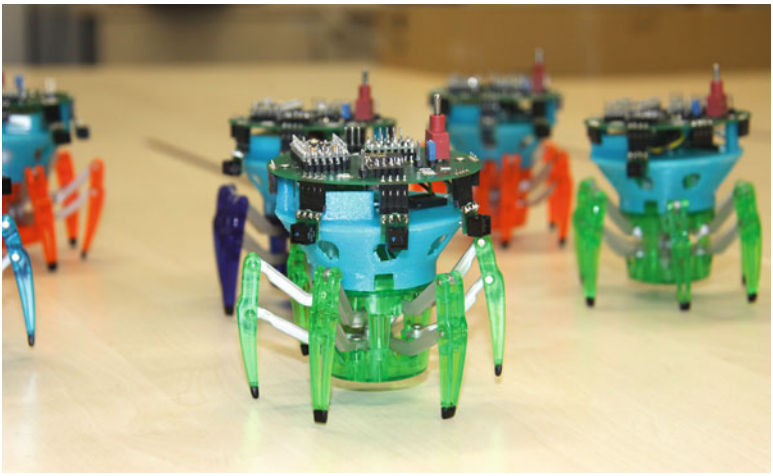


Fig. 2.16 A six-legged swarm robot with a diameter of 10 cm [113] (photo by Wilfried Elmenreich)

Fig. 2.17 The autonomous underwater vehicle MONSUN [300]

water layers of different temperatures. Also these underwater acoustic modems are usually rather expensive.

Although there is currently a lot of research on flying drones (quadrocopters), none of the projects seems to focus on self-organizing drone swarms that do not make heavy use of global information. A reason may be that it is very challenging to control multiple quadrocopters within a small space, creating a lot of air turbulences, while having all of them rely on their onboard sensors exclusively. What can be found in the literature are combinations of robot swarms on the ground and flying robots. Examples are the work by Chaimowicz and Kumar [68] and the project Swarmanoid [98].

2.6 Further Reading

There are many good books on mobile robotics. Arkin's Behavior-Based Robotics [13] and Probabilistic Robotics by Thrun et al. [381] are two important books. The book of Floreano and Mattiussi [122] is not focused on robotics but treats a rare combination of subjects, such as behavior-based robotics, evolutionary computation, learning, artificial neural networks, self-organization, and swarm robotics. Complementary books are Robot Programming by Jones and Roth [203] and even Braitenberg's Vehicles [52] although it does focus exclusively on behavioral aspects instead of robotics.

2.7 Tasks

2.7.1 Task: Kinematics of Differential Steering

The equations for the kinematics of differential steering were given in Eqs. (2.1) to (2.8). Use Eqs. (2.2), (2.4), and (2.5) to integrate x, y, and ϕ numerically over a time interval $t \in [0, 3]$ for the following situation. Wheelbase and initial position (without units) are set to

$$\begin{aligned} b &= 0.05, \\ x_0 &= 0, \\ y_0 &= 0, \\ \phi_0 &= 0. \end{aligned}$$

The velocities for the first third of the time interval ($t \in [0, 1]$) are set to

$$\begin{aligned} v_r &= 1.0 \\ v_l &= 0.9 \end{aligned}$$

At time $t_1 = 1$ and for the time interval $[1, 3]$ the wheel velocities are changed to

$$\begin{aligned} v_r &= 0.9 \\ v_l &= 1.0. \end{aligned}$$

Discretize time in steps of $\Delta t = 0.01$.

a. Calculate the robot's trajectory for the above setting using Eqs. (2.2), (2.4), and (2.5).

b. Let's assume that we have an error of $v_l \pm 0.01$ for the left wheel velocity (e.g., due to errors in the wheels or odometry, etc.). Calculate two additional trajectories for the whole time interval ($t \in [0, 3]$), one for $v_l + 0.01$ and one for $v_l - 0.01$, but using the integrated equations (2.6), (2.7), and (2.8) now.
c. Plot all three trajectories in one diagram and compare them.

2.7.2 Task: Potential Field Control

Implement a data structure for a potential field P (e.g., an array). Generate a potential field with a slope in one direction over the whole length (in Fig. 2.10, high potential on the left, low on the right), slopes leading away from the borders, and add a dozen of obstacles in the form of local maxima within the potential field (e.g., cone-shaped, Gaussian—as you wish). Place the robot randomly at a position with high potential at one side (in Fig. 2.10, robot is started on the left).

a. Generate trajectories for several potentials and robot placements with a controller that reacts directly to the gradient of the potential field, that is, the slope in x- and y-direction of the potential field at the robot's current position directly determines the robot's velocity v. The robot's displacement in one integration step is

$$\Delta x(t) = v_x(t)\Delta t,$$
$$\Delta y(t) = v_y(t)\Delta t,$$

and the velocity is determined by the gradient of the potential field:

$$v_x(t) = \frac{\partial}{\partial x}P(x, y),$$
$$v_y(t) = \frac{\partial}{\partial y}P(x, y).$$

An approximation of the gradient is obtained easily by subtracting two neighboring values of the potential field in your data structure $P[\cdot, \cdot]$:

$$\frac{\partial}{\partial x}P(x, y) \approx P[i, j] - P[i + 1, j],$$

for appropriate indices i and j. You have to play around with parameters to make it work: choose an appropriate integration step Δt and/or nominal velocity v, optionally normalize the slope of the potential field to one and scale it by a factor up or down, etc.
b. Generate trajectories for several potentials and robot placements with a controller that reacts indirectly to the gradient of the potential field, that is, the slope in x- and y-direction of the potential field at the robot's current position

only determines the acceleration of the robot. The robot's displacement is still determined by

$$\Delta x(t) = v_x(t)\Delta t,$$
$$\Delta y(t) = v_y(t)\Delta t,$$

but the robot's velocity is now determined by

$$\Delta v_x(t) = c v_x(t - \Delta t) + \frac{\partial}{\partial x} P(x, y),$$
$$\Delta v_y(t) = c v_y(t - \Delta t) + \frac{\partial}{\partial y} P(x, y),$$

for $0 < c \ll 1$ and followed by a normalization $\mathbf{v}/|\mathbf{v}| = (v_x, v_y)/|(v_x, v_y)|$ to prevent a too big increase of the velocity. That way the robot can have a kind of memory of former directions. Experiment with different parameters for the steepness of obstacles, normalize the acceleration vector, and implement a discount factor c that reduces the influence of former accelerations. What is a good setting such that the robot always (or at least most of the time) reaches the low values of the potential?

2.7.3 Task: Behaviors of a Single Robot

Design a simple environment, which is rectangular, bounded by walls, and almost empty (maybe a few obstacles). Initially, position the robot randomly somewhere in your robot arena.

a. Program a collision avoidance behavior—a robot that moves around while avoiding to crash into walls.
b. Program a wall follower—a robot that moves along the walls without touching them (i.e., avoid the easy solution of letting the robot slide along the wall).
c. Program an appropriate behavior for a vacuum cleaning robot. What are strategies to ensure that (almost) everything gets clean?
d. Just play around a little and see what kind of behaviors you can generate.

Chapter 3
Short Journey Through Nearly Everything

"And what happens to that incredibly complex memory bank that remembers the whole system during these periods of 'swarming'?"

—Stanisław Lem, The Invincible

"We take off into the cosmos, ready for anything: for solitude, for hardship, for exhaustion, death. [. . .] A single world, our own, suffices us; but we can't accept it for what it is."

—Stanisław Lem, Solaris

Abstract We do a little example tour through many methods and ideas we are going to study in this book.

This is a quick walk through the methodology that is of interest to design swarm robot systems. We model a robot controller with a finite state machine for a collective-decision-making problem. We immediately face the typical challenge of distinguishing between microscopic information that is available to an individual robot and macroscopic information that is only available to an external observer. We continue with a simple macroscopic model of collective-decision making and discuss whether it represents a self-organizing system.

3.1 Finite State Machines as Robot Controllers

A robot controller can be represented by a finite state machine with states being associated to actions and transitions triggered, for example, by sensor input or timers. A state represents constant actuation for the time spent in that state. An example for a collision avoidance behavior modeled by a finite state machine is given in Fig. 3.1. The robot has a sensor to the left s_l and a sensor to the right s_r.

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2_3

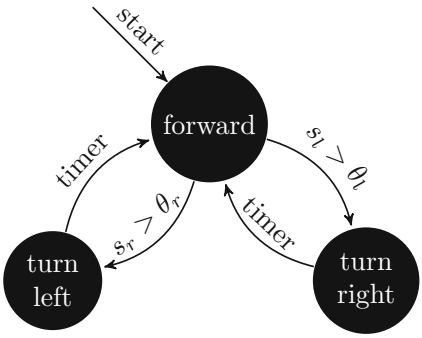


Fig. 3.1 Finite state machine for a collision avoidance behavior

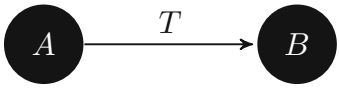


Fig. 3.2 Minimal example of a finite state machine

Thresholds θ_l and θ_r determine when an object is too close (bigger values of s indicate closer objects). The turns are executed for a defined time until a timer is triggered.

A minimal example is shown in Fig. 3.2. The condition of the transition T can depend on a certain sensor value and a threshold, on a timer, on a received message, etc.

3.2 State Transitions Based on Robot–Robot Interactions

In the following we focus on state transitions that depend on robot–robot interactions. In particular we focus on situations when the states of neighboring robots determine each robot's state transitions and hence its behavior. Say there are only two possible states: A and B. Say the swarm size is N. Then we can define a variable a that counts robots currently in state A and a variable b that counts robots in state B. Both a and b reflect global knowledge about the swarm that is available to an observer but not to the robots themselves. An obvious condition is $N = a + b$. Furthermore, we can calculate the fraction of robots that are in state A by $\alpha = \frac{a}{N}$.

We assume that an agent is able to determine the internal state of all its neighbors. This could be implemented by explicit messaging or, for example, each robot could switch on a colored LED which encodes its internal state and which could be detected by vision. Furthermore, we assume that all robots also keep moving around although we do not specify a particular purpose of doing so here. However, if the robots are in motion, then their neighborhoods are also dynamic.

Based on the unit disc model we assume a sensor range r. For a robot R_0 all robots within distance r are within its neighborhood $\mathcal{N}$. For the situation shown in Fig. 3.3 we have $\mathcal{N} = \{R_1, R_2, R_3, R_4\}$. We assume that R_0 knows the states of all

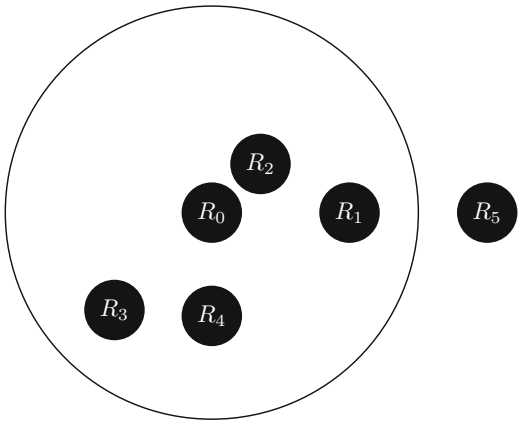


Fig. 3.3 Neighborhood of robot R_0 is $\mathcal{N} = \{R_1, R_2, R_3, R_4\}$

neighboring robots $\mathcal{N}$. In similarity to the above defined variables we can introduce variables $\hat{a}$ and $\hat{b}$ that give the number of neighboring robots of robot R_0 plus itself that are in state A and B, respectively. We have $|\mathcal{N}| + 1 = \hat{a} + \hat{b}$ and we can give a fraction of robots that are in state A by $\hat{\alpha} = \frac{\hat{a}}{|\mathcal{N}|+1}$.

In the following we assume that a robot's state transitions between A and B depend exclusively on $\hat{\alpha}$. For example, we could say if the robot is currently in state A and it measures $\hat{\alpha} < 0.5$ (i.e., there are more neighbors in state B than state A) then it switches to state B.

3.3 Early Micro-Macro Problems

We can interpret the swarm fraction $\hat{\alpha}$ as a measurement by robot R_0 of the actual current global situation which is given by swarm fraction α. This is also called "local sampling" and will be discussed in detail later (see Sect. 5.2). Generally we have $\alpha \neq \hat{\alpha}$. Under which conditions could we hope for $\alpha \approx \hat{\alpha}$? In average and for a so-called well-mixed system, that is a system without a bias, we can assume $\alpha \approx \hat{\alpha}$. However, the well-mixed assumption does often not hold and the sampling error (i.e., variance in the robot's measurements) can have systematic effects and hence introduce a bias as well. This complex of problems is already a small taste of the micro-macro problem that will be discussed later in detail. The main challenge of swarm robotics is to find connections between the microscopic level (here, local measurements of $\hat{\alpha}$) and the macroscopic level (here, the actual global situation α). We also speak of establishing a micro-macro link.

3.4 Minimal Example: Collective Decision-Making

Next, we investigate a minimal example of collective decision-making (more details later, see Chap. 6). The task in collective decision-making is typically to find a consensus, that is, 100% of the robots in the swarm agree on a decision which could, for example, be to switch to the same internal state. Based on the above defined result $\hat{\alpha}$ of locally sampling the neighborhood and a threshold of 0.5 we can define a finite state machine for this little collective decision-making scenario (see Fig. 3.4).

These transition rules define what we call a majority rule because this approach tries to reinforce the current majority. If there are more close-by robots in state A, then the considered robot switches to A (otherwise it stays in B). If there are more close-by robots in state B, then the considered robot switches to B (otherwise it stays in A). Our hope is that in average the local measurement $\hat{\alpha}$ gives a good approximation to the global state α (cf. micro-macro problem). As a consequence, the whole swarm should converge on a consensus of either $\alpha = 1$ or $\alpha = 0$.

3.5 Macroscopic Perspective

On the microscopic level the situation is relatively clear. The measurement of $\hat{\alpha}$ is probabilistic but the state switching behavior for a given $\hat{\alpha}$ is deterministic. If we want to determine the macroscopic effect of this microscopic behavior, then things get a bit more difficult. We have to look into the combinatorics of all possible neighborhoods. For simplicity we restrict ourselves to a small neighborhood of $|\mathcal{N}| = 2$ and we make use of the above defined true swarm fraction α.

The probability $P_{B\to A}$ to switch from B to A is given by

$$P_{B\to A}(\alpha) = (1-\alpha)\alpha^2, \tag{3.1}$$

because the considered robot has to be in state B (probability $1-\alpha$), we assume that its neighborhood is statistically independent, so we multiply with the probability that both ($|\mathcal{N}| = 2$) neighboring robots are in state A (otherwise the transition condition $\alpha > 0.5$ is not satisfied). According to combinatorics there would be $\binom{3}{1} = 3$ ways of arranging two A and one B (BAA, ABA, and AAB) but only the first one results in a switch with the considered robot in state B. A plot of Eq. (3.1)

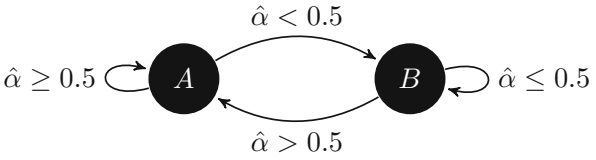


Fig. 3.4 Minimal example of collective decision-making based on the majority rule

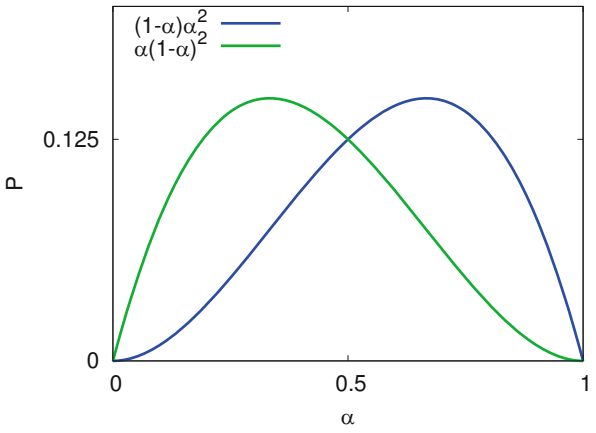


Fig. 3.5 Switching probabilities, Eq. (3.1)

is shown in Fig. 3.5. For $\alpha > 0.66$ there are too few robots in state B that could potentially switch; that is why the probability is decreasing with increasing α. For $\alpha < 0.66$ there are too few robots in state A that could generate a local majority; that is why the probability is decreasing with decreasing α.

The probability to switch from A to B is fully symmetric and defined by

$$P_{A\to B}(\alpha) = \alpha(1-\alpha)^2. \tag{3.2}$$

In all other situations we observe no switch and we get

$$P_{A\to A}(\alpha) = \alpha^3 + 2(1-\alpha)\alpha^2, \tag{3.3}$$

whereas the term $2(1-\alpha)\alpha^2$ accounts for the above-mentioned situations ABA, and AAB where we would have a majority of A but the considered robot is already in state A and hence does not switch to A. In symmetry we get

$$P_{B\to B}(\alpha) = (1-\alpha)^3 + 2(1-\alpha)^2\alpha. \tag{3.4}$$

3.6 Expected Macroscopic Dynamics and Feedbacks

Now we would like to know the expected macroscopic dynamics of the system, that is, how does the swarm fraction α develop over time. For that we need to introduce a representation of time. We define a time interval Δt that is long enough to observe state transitions and short enough to observe not too many state transitions. We define the expected change $\Delta\alpha$ of α by using the above defined probabilities

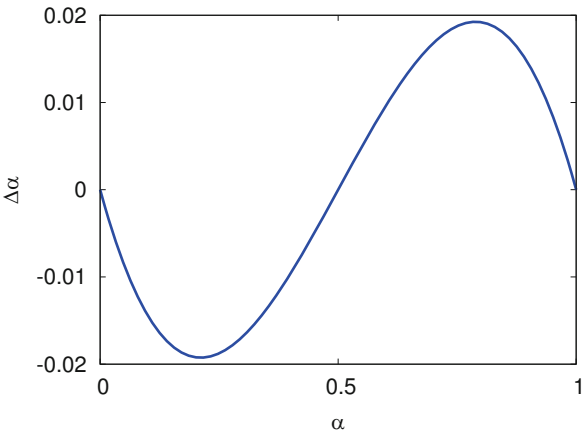


Fig. 3.6 Expected average change $\Delta\alpha$ of α

of state transitions $P_{B\to A}$ and $P_{A\to B}(\alpha)$ and by weighting them according to the contribution of the state switches to the change of $\Delta\alpha$

$$\frac{\Delta\alpha(\alpha)}{\Delta t} = \frac{1}{N}((1-\alpha)\alpha^2) - \frac{1}{N}(\alpha(1-\alpha)^2). \tag{3.5}$$

The first term represents transitions $B \to A$, which contribute positively to α (i.e., generating more robots in state A). The second term represents transitions $A \to B$, which contribute negatively to α (i.e., generating more robots in state B). Factor $1/N$ accounts for an assumed switching rate of one robot per time step. Equation (3.5) is plotted in Fig. 3.6.

What is shown in Fig. 3.6 represents a feedback process. Whether it is positive or negative feedback is easily determined visually. The left half of the diagram represents minority situations in terms of state A because we have $\alpha < 0.5$ there. In the left half we also have only negative values for $\Delta\alpha$. That means the minority of state A is even reinforced. Similarly, in the right half of the diagram we have majority situations in terms of state A because $\alpha > 0.5$. In the right half we also have only positive values for $\Delta\alpha$. That means the majority of state A is even reinforced. Hence, we have positive feedback.

Does this robot swarm qualify as self-organizing system? It contains positive feedback. Does it also contain negative feedback? It is not obvious but it does contain negative feedback. All real-world systems have limited resources, so also here the positive feedback is stopped once the minority is eaten up and no robots remain to switch to the majority (for $\alpha = 0$ and $\alpha = 1$). The system has initial fluctuations that determine whether we end up with 100% of the robots in state A or 100% in state B, and we have multiple interactions between many robots. The balance between exploitation and exploration is, however, biased towards exploitation. Once the swarm has converged on a consensus it will never leave it. This can be bad in certain situations, for example, in dynamic environments when

the swarm should stay adaptive to new situations. Hence, this robot swarm is close to a self-organizing system but misses exploration.

How could exploration be included? We could allow robots to spontaneously switch between states. Say in each time step they have a 5% chance of a spontaneous switch. The above given expected change $\Delta\alpha$, Eq. (3.5), then changes to

$$\frac{\Delta\alpha(\alpha)}{\Delta t} = \frac{1}{N}((1-\alpha)\alpha^2) - 0.05\alpha - \frac{1}{N}(\alpha(1-\alpha)^2) + 0.05(1-\alpha). \tag{3.6}$$

While $\alpha = 0$ and $\alpha = 1$ were fixed points before, they are not anymore. For $\alpha = 0$ we have now $\frac{\Delta\alpha(\alpha)}{\Delta t} = 0.05$ and for $\alpha = 1$ we have $\frac{\Delta\alpha(\alpha)}{\Delta t} = -0.05$. Hence, we have regions of negative feedback at the two boundaries for about $\alpha < 0.052$ and $\alpha > 0.948$. As a consequence we always have a few robots in the opposite state, who can serve as explorers to check whether the other option has improved in utility. So then we have a self-organizing system.

3.7 Further Reading

This chapter is meant to give you a small taste of what we are going to discuss in this book. So you can just keep reading or if you want to know many more details about modeling collective-decision-making systems then you can continue by reading Valentini's book on the subject [392]. Vigelius et al. [402] study micro-macro models of collective decision-making. Couzin et al. [76] explain how animals find good decisions. Biancalani et al. [41] describe a very simple decision-making model that, however, most likely does not scale. Reina et al. [324] give a hint on how a software engineering approach for collective decision-making in swarm robotics can look like.

3.8 Tasks

3.8.1 Task: Plot the Macroscopic Dynamic System Behavior

Plot the expected change $\frac{\Delta\alpha(\alpha)}{\Delta t}$ with negative feedback as given in Eq. (3.6). How does it change when you choose different probabilities for spontaneous switching?

3.8.2 Task: Simulate Collective Decision-Making

Write a little simulation that simulates non-embodied agents moving randomly on a torus and switching states according to the rules we have defined in this chapter.

Monitor the current global state α and plot its trajectory, that is, how it changes over time. Check also different probabilities for spontaneous switching again. Does it appear that the swarm changes from large majority in favor of option A to option B and vice versa? How long does it take?

Chapter 4
Scenarios of Swarm Robotics

All this time we've been gazing upwards in anticipation of an alien species arriving from space, when intelligent life-forms have been with us all along, inhabiting part of the planet that we've never seriously attempted to explore.

—Frank Schätzing, The Swarm

From what you say it follows that robots should be constructed quite differently from the way we've been doing it in order to be really universal: you'd have to start with tiny elementary building blocks, primary units, pseudo-cells that can replace each other, if necessary.

—Stanisław Lem, The Invincible

Abstract We do an extensive check of what typical scenarios of swarm robotics have been investigated and what methods have been published.

This is an extensive guide through the literature on swarm robotics. It is structured by the investigated scenarios and starts from tasks of low complexity, such as aggregation and dispersion. A discussion of pattern formation, object clustering, sorting, and self-assembly follows. Collective construction is already a rather complex scenario that combines several subtasks, such as collective decision-making and collective transport. We take the example of collective manipulation to discuss the interesting phenomenon of super-linear performance increase. Not only the swarm performance increases with increasing swarm size but even the individual robot's efficiency. Flocking, collective motion, foraging, and shepherding are discussed as typical examples of swarm behaviors. Bio-hybrid systems as combinations of robots and living organisms are quickly introduced. We conclude with a discussion of what could arguably be called "swarm robotics 2.0"—a few recent very promising approaches, such as error detection, security, swarms as interfaces, and swarm robotics as field robotics.

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2_4

4.1 Aggregation and Clustering

Aggregation is one of the most fundamental swarm behaviors frequently observed also in natural swarms (see Fig. 4.1). Not only but especially for natural swarms, it is important to stay together. Otherwise the swarm would separate into several parts, decrease in size, and eventually its survival may be endangered. Aggregation behaviors are observed in many natural swarms, such as young honeybees [372], ladybugs [229] (see Fig. 4.1, top left), and monarch butterflies [412, see Fig. 4.1, lower left]. In aggregation the task for the robots is to position themselves close to each other in one spot. This is achieved by minimizing the distances between robots. The actual position of the aggregation spot may be specified or unspecified. If it is unspecified, then the robot swarm self-organizes to find a consensus on its aggregation position. Hence, aggregation at an unspecified spot has an inherent collective decision-making component. The aggregation spot can be specified, for example, by saying the swarm should gather at the brightest or warmest spot. Then each robot, possibly in collaboration with others, has to find that position and stop there.

The naive control approach to aggregation in swarm robotics is to move in a random direction, once you see another robot you stop. Although initially robots may reduce the average distance between robots, the swarm system gets caught

Fig. 4.1 Aggregations of insects (top left photo of ladybugs by Katie Swanson (Nature ID blog) with permission, other photos are CC0)

in a local optimum immediately. Most clusters of robots are small, maybe two or three robots mostly. Even in this simple problem of aggregation, we face already a challenging problem of exploitation of local knowledge and exploration of the environment. Robots should not greedily exploit knowledge about their environment by staying stopped close to the first robot they meet. Instead they should also explore their surrounding sometimes and check whether bigger clusters exist that they should join.

Also note that a generic optimal solution to this exploration–exploitation tradeoff does not exist because it depends on the respective scenario. How often an explorative behavior is useful depends on the swarm density (robots per area), the swarm size, and the robot's speed. In addition, the robots are usually not synchronized, certain behaviors may create deadlocks, and the robots can hardly know when the exploration phase in the swarm is over. Finally, in the case of aggregation at a specified spot, the environment may be dynamic. Hence, the robots should always stay explorative and send out scout robots that check whether a better aggregation spot has emerged at a different location.

The literature about aggregation in swarms is rich. The most popular algorithm is, as mentioned before, BEECLUST [344] originally published by Schmickl et al. [349] and Kernbach et al. [212]. For details about BEECLUST, see Chap. 7. BEECLUST was analyzed by Hereford [187] and extended with a fuzzy-based approach by Arvin et al. [17, 19]. The phenomenon of symmetry breaking in BEECLUST, when the swarm is presented two options (locations) of the same utility, was investigated by Hamann et al. [168, 171]. Symmetry breaking may seem a simple problem. The swarm has to find consensus about options A and B. Both options have identical utility. A human being faced with such a problem randomly picks one. The swarm, however, has to collectively decide about which one of the two to choose based on local information only. This problem corresponds directly to the problem of having two clusters of robots with the aim of merging them. Who stays and who moves to the other cluster? This is an even more difficult question if the other cluster is out of the robots' perception range. This is again an exploitation–exploration tradeoff that cannot easily be solved. Depending on the environmental conditions and the robot's capabilities (e.g., speed and sensor range) in the given scenario, appropriate probabilities of leaving and joining a cluster need to be defined. Joining and staying at bigger clusters should be more desirable, hence, creating a positive feedback effect (i.e., big clusters tend to get bigger).

A quite different work is that of Correll et al. [75]. They report a technological system to aggregate cattle via social control. The idea is to exploit the adaptive behavior of cows and control only a fraction of the herd. Some cows follow other cows, hence, not all need to be controlled directly. The device is carried by the cows and triggers an undirected stimulus when they enter undesired areas, hence, implementing a "virtual fence."

Finally, for a theoretical approach to collective behaviors and aggregation specifically you can read the paper of Chazelle [69] and his other works. Popkin [313] just starts from flocking and then goes into a rather philosophical but inspiring discussion of whether there is "the physics of life."

4.2 Dispersion

Dispersion is the behavior directly opposite to aggregation. A robot is supposed to position itself as far as possible from every other robot but while staying in contact. The distances between robots should be maximized but the robots should stay within communication range. This is a useful behavior if large areas need to be monitored or if the swarm should minimize its robot density. The primarily expected spatial distribution of robots is a uniform distribution, that is, the robots position themselves in a regular pattern. The task can, however, also be approximated by a random uniform distribution, or even clusters may be acceptable depending on the underlying task. This corresponds well to the three types of population distributions in ecology: clumped, random, and regular (see Fig. 4.2).

An implementation is reported by Payton et al. [304] in a popular paper on so-called pheromone robotics. The term "pheromone" only serves as a buzzword here and should not be interpreted as indicating an appropriately bio-inspired pheromone approach. Instead the robots communicate by infrared messages and measure distances between robots by measurements of signal strength only. A similar approach is reported by Ludwig and Gini [242] but focuses on radio communication. The study is, however, only done in simulation.

Also McLurkin and Smith [262] study in robot experiments with 56 robots how a swarm of robots can disperse efficiently concerning time and forming a uniform distribution. They use a gradient-based multi-hop messaging protocol. Their algorithm alternates between two sub-routines with one spreading the robots evenly and the other one directing robots towards unexplored areas.

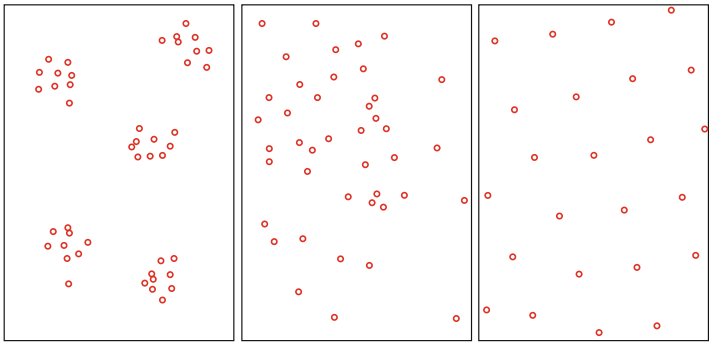


Fig. 4.2 In ecology there are three types of population distributions: clumped, random, and regular. (GNU Free Documentation License)

4.3 Pattern Formation, Object Clustering, Sorting and Self-Assembly

Once we go from aggregation behaviors one step further in complexity, we talk about pattern formation, clustering and sorting of objects, and self-assembly of robots. Pattern formation means robots aggregate in defined shapes or shape their environment. In clustering robots move objects around to aggregate the objects or to sort them according to features, such as color, size, or age. Self-assembly means robots physically connect to each other to form more powerful super-robots.

4.3.1 Pattern Formation

Once we go beyond mere aggregation, robots may be asked to form certain patterns, to sort themselves, to self-assemble into a super-robot, or to sort objects. Self-organized pattern formation is also found in many natural systems [380], predominantly in biology but even in self-organizing stone and salt patterns [213]. Many of these patterns can be explained by reaction–diffusion mechanisms and activator–inhibitor schemes [26]. Although Turing [389] published his fundamental paper about pattern formation already in 1952, as late as in the 1970s biologists became aware of how biological pattern formation may be explained by self-organization [144]. Well-known phenomena such as animal coat markings [289, 290] and patterns in insect embryogenesis [205] were addressed. Another popular example are the pigmentation patterns of shells [265, 268]. More relevant for swarm robotics are, for example, the *Physarum* transport networks [201, 202] and pattern formation in simple, simulated agent swarms [170].

Besides the obvious colorful patterns, such as animal coat markings, there are also spatio-temporal patterns. The formation of a certain traffic flow in pedestrians, where people walk in one direction on the right half and in the opposite direction on the left half of the path, is also an example of self-organized pattern formation. So there is collective behavior in vertebrates and especially also in human beings [78]. Often people ignore the paved system of walkways, take shortcuts, and create new trail systems [185, 186]. A special example of pattern formation are the trail networks formed by ants. It is well-known that ants use stigmergy to organize their foraging behavior, that is, they mark trails with pheromones. The navigation within these trail networks seems to be based on the specific geometry of the trail system. In particular, defined angles of trail bifurcations can be leveraged by the ants to know which direction leads to food and which to the nest [194]. This approach can also be used in swarm robotic systems for orientation in a trail network [173]. However, then one requires a feasible emulation of pheromones, such as proposed by Mayet et al. [259]. Alternatively, one can use heat trails [336].

Roughly following the idea of trail networks, Nouyan et al. [294] implemented a robot swarm system that positions robots in chain formations. These robot chains can, for example, be used in a foraging scenario to guide another robot along the chain to the food.

4.3.2 Clustering

Clustering of objects was often investigated in simulation but there are also many hardware implementations [48]. The classical termite simulation model of Resnick [326] resembles a simple behavior of termites that collect wood chips and cluster them. The two only rules are: if you do not have yet a wood chip, then pick one up randomly; if you have a wood chip already, place it next to a randomly chosen wood chip. As refinement one can introduce probabilities, such that it is less likely to pick up a wood chip positioned close to other wood chips and more likely to place a wood chip close to many other wood chips.

An interesting alternative work on object clustering, as mentioned in Sect. 1.5, is the approach of "anti-agents" by Scheidler et al. [341] and Merkle et al. [272]. The standard approach to object clustering is probabilistic with a probability P_{pick} to pick up an object and P_{drop} to drop it. An anti-agent is a robot that behaves differently than the majority of robots. Scheidler et al. [341] investigate different choices of anti-agents, such as random behavior, inverted neighborhood (i.e., inverted perception of the neighborhood), and reverse anti-agents. Reverse anti-agents invert the above probabilities to $1 - P_{\text{pick}}$ and $1 - P_{\text{drop}}$. Scheidler et al. [341] found a nonintuitive effect, namely that certain numbers of reverse anti-agents increase the observed clustering effect. This could lead to a concept of increasing the efficiency of swarm systems with appropriate numbers of anti-agents.

4.3.3 Sorting

Once there are multiple types of objects that need to be clustered we are entering the domain of sorting. For example, there is multi-object sorting in "annular structures" by Wilson et al. [417]. Annular structures are defined as forming a central cluster of one class of objects, and surrounding it with annular bands of the other classes, each band containing objects of only one type. This research is inspired by ants (*Leptothorax albipennis*) that build a perimeter wall around their nest [128]. A related behavior is brood sorting in ants [127]: "Different brood stages are arranged in concentric rings in a single cluster centred around the eggs and micro-larvae."

A similar task is patch sorting [270]. Patch sorting is defined as "grouping two or more classes of objects so that each is both clustered and segregated, and ... each lies outside the boundary of the other" [188]. Melhuish et al. [270] implemented an appropriate behavior on six robots that successfully sorted 30 frisbees (three types) into three patches.

Alternatively to sorting objects one can also sort robots directly. Say there are two types of robots in the swarm, red and blue robots, and there are two aggregation areas. The swarm is supposed to collectively decide about which area belongs to the red and which area to the blue robots. Then they should aggregate there and hence sort themselves [93].

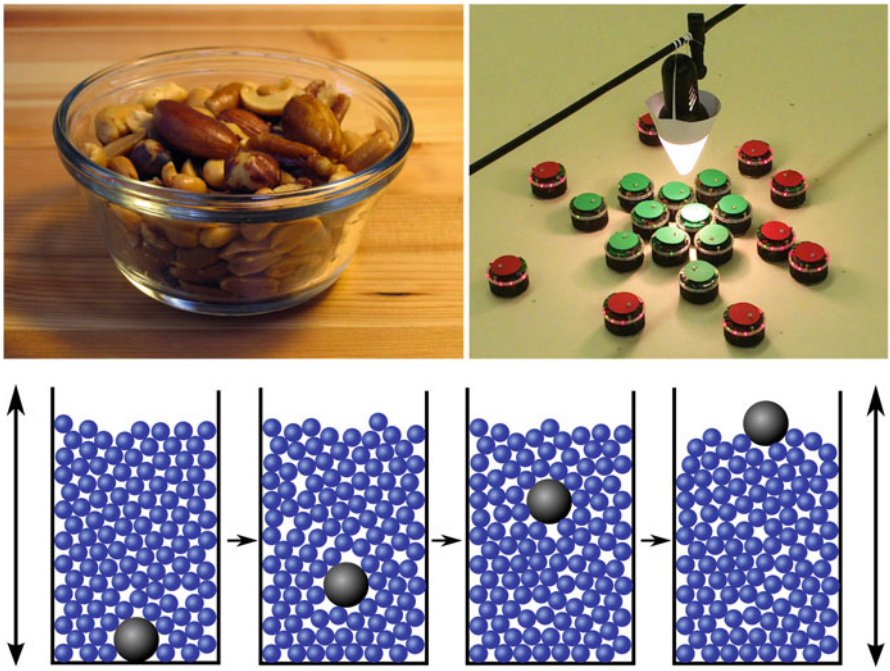


Fig. 4.3 The Brazil nut effect as inspiration for robot segregation (photo of nuts by Melchoir, CC BY-SA 3.0; photo of robots by Roderich Groß with permission [70]; schema by Gsrdzl, CC BY-SA 3.0)

A particularly remarkable study of swarm robot sorting was done by Groß et al. [155] and Chen et al. [70]. They studied a segregation behavior in a swarm of robots based on the so-called Brazil nut effect [280]. The Brazil nut effect is the unfortunate phenomenon of self-organized sorting in breakfast cereals. The big Brazil nuts gather at the top and all the small nuts end up at the bottom of the bucket. Obviously a dedicated cereal eater does not want to eat big nuts only on Monday and exclusively tiny nuts on Friday. Self-organized sorting in nuts works only if energy is added to the system, for example, by vibrations during the transport of the cereals in a truck. All nuts jump up and down but small nuts gather in the cavities below big nuts before they can fall down again (see Fig. 4.3, bottom). Hence, the big nuts move up and the small nuts move down. Groß et al. [155] and Chen et al. [70] cleverly transferred this self-organizing, physical effect to the domain of swarm robotics. Although the robots are physically homogeneous unlike the nuts, they are programmed to avoid collisions with different minimal distances. The robots have different collision avoidance radii, which corresponds directly to nuts of different size. As an effect, once the robots are asked to aggregate at a specific spot (e.g., approaching a light, Chen et al. [70]), robots with small collision avoidance radius gather in the middle and robots with big collision avoidance radius gather at the boundaries (see top right photo in Fig. 4.3, green robots have a smaller collision avoidance radius than red robots).

4.3.4 Self-Assembly

"Self-assembly can be defined as a process by which pre-existing discrete components organize into patterns or structures without human intervention" [154]. In contrast to the above pattern formation, the idea is to physically connect smaller autonomous parts to construct a bigger entity that has relevant physical or architectural function, such as "bivouacs, bridges, curtains, droplets, escape droplets, festoons, fills, flanges, ladders, ovens, plugs, pulling chains, queen clusters, rafts, swarms, thermoregulatory clusters, tunnels, and walls" [9]. Self-assembly is relatively common in insects [9].

The s-bot of the swarm-bots project (see Sect. 2.5.1) was developed to allow for self-assembly [101]. Groß and Dorigo [154] use self-assembly with s-bots to form chains of robots that can pull a heavy object collectively (i.e., collective transport, see Sect. 4.5). The robots can physically connect to each other with a gripper, coordinate their motion, and pull collectively. In similar studies, the authors show how groups of self-assembled robots can cross gaps and steep hills, that they could not have crossed as single robots [296, 387].

The concept of self-assembly in robotics is taken to a next level in reconfigurable modular robotics. The idea is to design autonomous robot modules that can physically connect and self-assemble into super-robots (cf. the quote of Stanislaw Lem at the beginning of this chapter). Rubenstein and Shen [333] give an overview based on simulation of what is theoretically possible following this concept. The full power of this approach is available once the robot modules can autonomously reconfigure the shape online at runtime. Then the robot system has levels of robustness similar to swarm robots because it can self-repair [11, 12, 365]. Self-reconfigurable modular robotics with aspects of swarm robotics was investigated in the projects SYMBRION and REPLICATOR [172, 240, 345, 348]. The robots can dock to each other and the docking allows to physically connect, to share energy, and to establish a communication bus. The control of reconfigurable modular robots is considered to be challenging, especially if also the self-assembled shapes and topologies of the super-bot are allowed to be adaptive. Not all shapes can be anticipated before and the control needs to deal with them at runtime [169, 432].

The Kilobot (see Sect. 2.5.4) was used by Rubenstein et al. [332] to emulate self-assembly by merely positioning the robots next to each other (see Fig. 4.4). The robot swarms in these experiments are the biggest reported until now with up to 1024 Kilobots. The task for the robot swarm is to position all robots properly to form predefined shapes. A challenge is to locate robots based on local information only. Rubenstein et al. [332] use four seed robots as anchor (see Fig. 4.4h) to form a gradient as coordinate system. The robots then move along the edge of the already formed aggregation in a line, which has a strong impact on the time overhead as waiting times scale linearly with the swarm size (see Fig. 4.4i). A problem that was addressed later by inverting the whole approach, that is, doing self-organized disassembly [140]. Similarly the problem of error propagation using the gradient approach was investigated [141].

Another example of self-assembly using Kilobots is presented by Divband Soorati and Hamann [94]. They present an adaptive strategy to form only

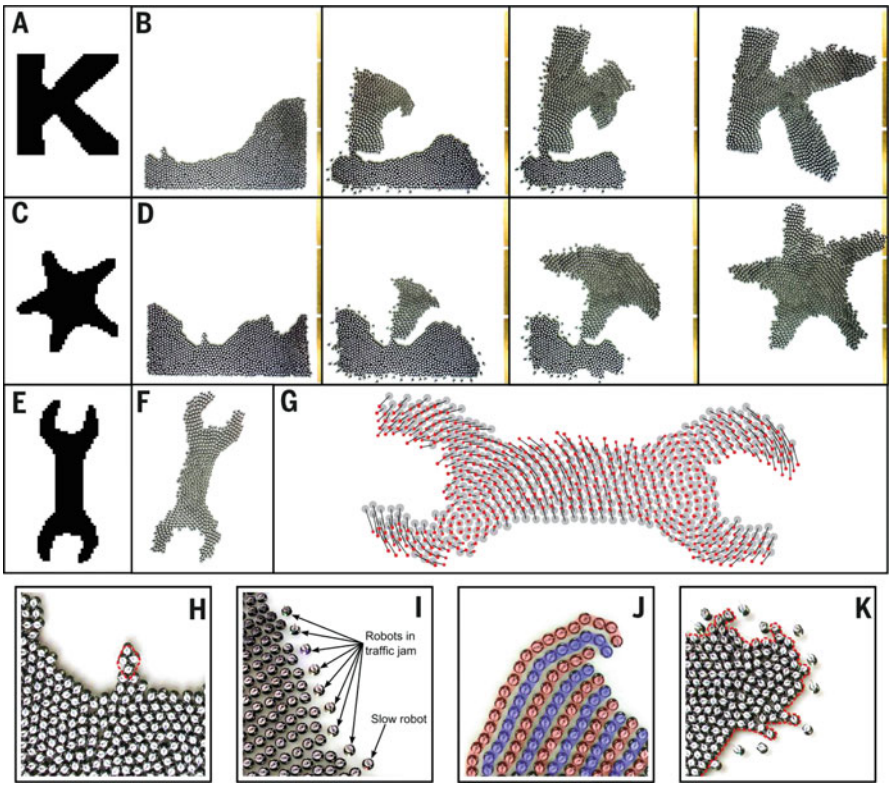


Fig. 4.4 Emulation of self-assembly with Kilobots [332] (printed with permission) (**a**, **c**, **e**) desired shape, known by each robot (**b**, **d**) self-assembly process from initial configuration to final configuration (**f**) final shape slightly bent due to errors in the process (**g**) accuracy of the final shape is measured by comparing actual to desired robot positions (**h**) four seed robots form an anchor (**i**) robots move along the edge in a line (**j**) robots in each of these rows share the same gradient value (**k**) erratic boundary due to moving robots pushing stopped robots

qualitatively predefined tree-structures. The trees are specified, for example, by their branching factor (i.e., bushiness of the tree). A related model that can be used in swarm robotics to control the self-assembly of growing tree structures is the so-called Vascular Morphogenesis Controller [434].

4.4 Collective Construction

Using a robot swarm to collectively construct buildings or any other architectural artifact is probably an intuitive idea. Construction is a task that seems easily paralellizable. In addition, collective construction may incorporate several other basic behaviors of swarm robotics, such as collective decision-making (e.g., where to start construction) and task partitioning (e.g., "I place my block here and you there").

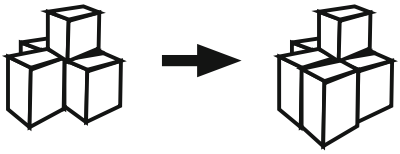


Fig. 4.5 An example rule of a cellular automaton in 3-d with a neighborhood of $3 \times 3 \times 2$ blocks

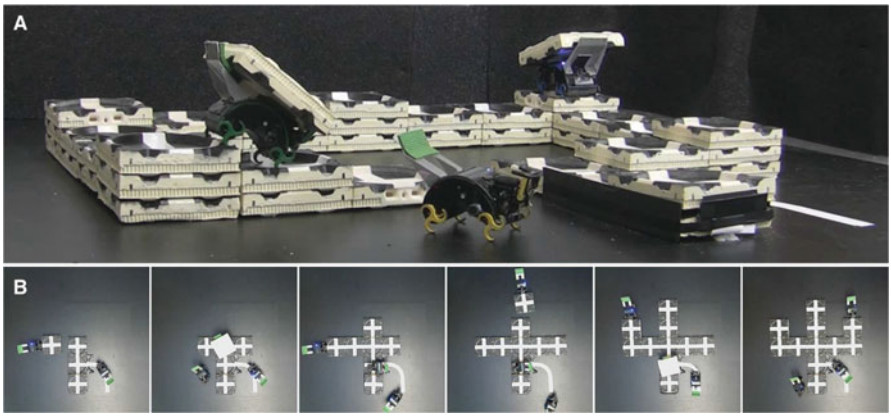


Fig. 4.6 Example of a robot swarm collectively constructing an architectural artifact [413] (printed with permission)

Collective construction is even more fascinating once we investigate how animals build their homes [177, 404]. In particular we are interested in building behaviors of social insects, such as honeybees, wasps, ants, and termites. They operate based on local information and in a self-organized way. An important work in this context are the papers by Theraulaz and Bonabeau [377, 378]. They model the building behavior as cellular automata, that is, the world is discretized as a lattice with cells (see Fig. 4.5). The animal looks only at its directly surrounding environment (e.g., where is building material and where are holes) and follows simple rules that map the state of the local environment to actions, such as "move forward" or "place material here." The major challenge for this cellular-automaton approach but also for any kind of collective construction is to break down the global blueprint (i.e., how will the "house" look like eventually) to simple local rules (i.e., if your neighborhood looks like this, then place material here). This approach is inherently based on stigmergy and it is unclear whether it is always possible to derive local rules for any given global blueprint. Even more so, it is unclear how that can be done efficiently.

The most popular paper on swarm construction was published by Werfel et al. [413]. They designed a dedicated robot platform that picks up flat blocks, puts them on its back, climbs stairs built of these blocks, and places the block at an appropriate position (see Fig. 4.6). Even for a single robot the challenge is to ensure that it does not place building blocks at positions that may result in deadlocks. For example, a robot may not be able to climb down the stairs anymore to pick up the next block. Once we allow multiple robots to construct in parallel, additional

deadlock situations may arise because robots block each other's way. Using building blocks also requires relatively high precision in all actions of the robot, ranging from picking up blocks, to placing them, and not falling off the construction.

Another example of an approach based on building blocks by Allwright et al. [5, 6] is currently under development. The robot is equipped with a forklift and moves "smart blocks" around. The robot can change the RGB LEDs of these smart blocks via radio to implement even more possibilities for stigmergy. For example, the colors can encode different levels of the construction.

Augugliaro et al. [20] report the art installation "Flight Assembled Architectures," a swarm of quadrotors to construct architectural artifacts. As an example, they build a 6-m-tall tower out of foam modules. The system makes use of global information, a central element, and robots are controlled offboard. There is an overhead motion capture system, a 19-camera Vicon T-40 system, to provide high-accuracy position information. The information and the robots' destinations are processed with a complex trajectory planning and trajectory generation algorithm.

A quite different approach was proposed by Heinrich et al. [184]. This concept of collective construction operates with continuous building material in the form of strips or strings that are braided together (see Fig. 4.7). Initially it may seem an unnecessary limitation to work on braids but the possibilities are actually almost infinite [399].

In Fig. 4.7a the braiding approach with mobile robots is shown. The applied robot is the Thymio II robot. Robots collectively braid, one strip each by a simple line-following behavior combined with a collision avoidance behavior. The resulting braid is shown on the right-hand side in Fig. 4.7a. In Fig. 4.7b the inspiration of this work is shown, a traditional May dance as practiced, for example, in Germany and Austria. For a short overview of the literature on self-organized construction, see the review paper of Gerling and Mammen [143].

4.5 Collective Transport

Collective transport is a good example for a scenario with potential for an actual swarm effect. A few robots may not be able to move a heavy object even for one centimeter but once a certain threshold in the swarm size is reached they can move it for any distance. Such an extreme steep increase in swarm performance (super-linear performance increase, cf. Sect. 1.2.1) can be observed only in few scenarios. This difference as a discrete jump in performance from zero to infinity exists basically due to the physical effects of stiction (friction of stationary objects) and kinetic friction (friction of moving objects). Otherwise there should be a continuous increase in the covered distance of the transported object with increasing swarm size.

There are a number of implementations using robots. Kube and Bonabeau [222] investigate box-pushing inspired by ant behaviors. They discuss a number of issues, such as how to overcome stagnation during the pushing action and taxis-based

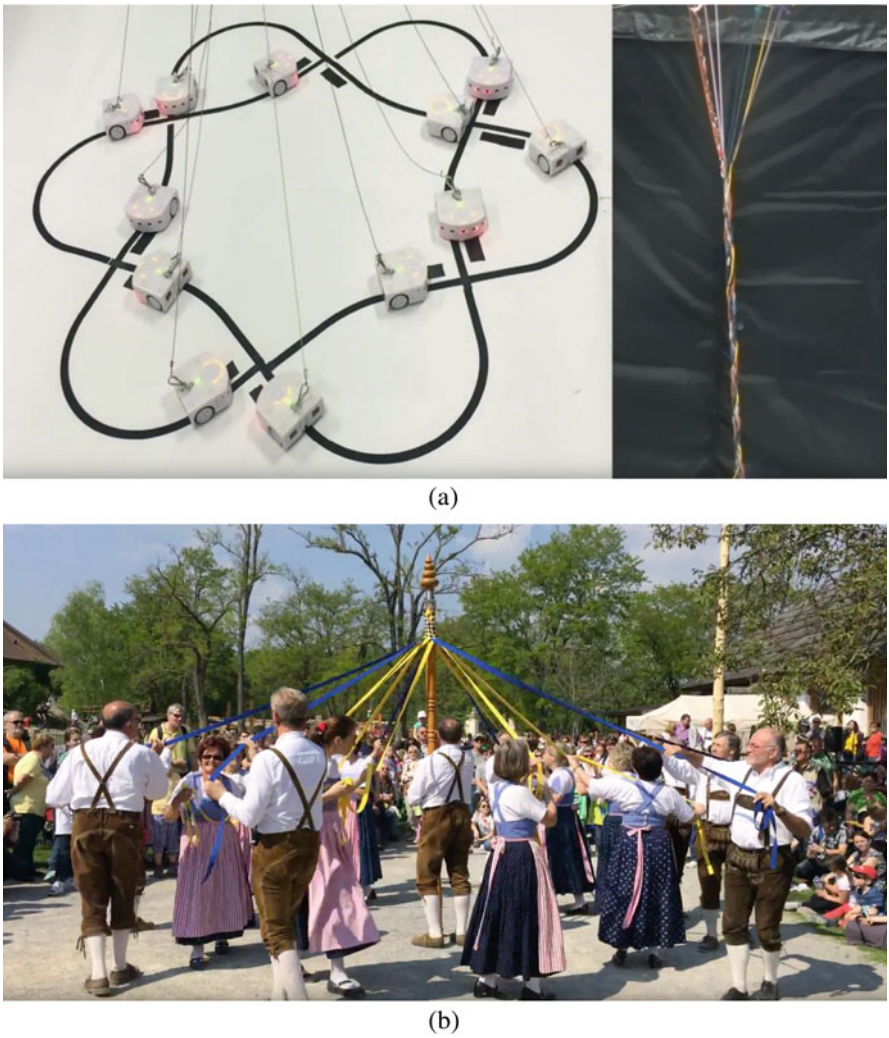


Fig. 4.7 Swarm construction by braiding continuous material [184]. (**a**) Braiding Thymio robots based on line following and resulting braid (**b**) Traditional May dance (photo by Martin Ladstätter, with permission)

transport, that is, pushing the box towards a light. Coordinating the actions of the robots based on local interactions and making use of appropriate sensors (e.g., touch sensor, proximity sensors, light sensor).

A similar behavior but including self-assembly is the above-mentioned work of Groß and Dorigo [154]. Nouyan et al. [295] make use of the same behavior but in the context of a rather complex foraging scenario where the object needs to be collectively transported along a chain of robots pointing the way back home.

Berman et al. [40] report an in-depth analysis of biological data of collectively transporting ants, its application to a robot control strategy, and its verification in simulation [418]. Habibi et al. [160] report a scalable distributed path planning algorithm for transporting a large object through an unknown environment. The area is sampled and the distributed Bellman-Ford algorithm [124] is used to build a shortest-path tree. The robot experiment was done with eleven "r-one" robots (see Sect. 2.5.5) [261].

Exciting opportunities arise when we consider quadrotors as option for collective transport. Mellinger et al. [271] report their approach to grasping and transporting building material in groups of quadrotors. Given that this is one of the earliest works focusing on flying robots for collective transport, this is certainly a big step forward. A drawback though is that an indoor-GPS system is used and the robots receive control values from a central computer. The research on actual swarms of quadrotors is still in an early stage because it is extremely challenging to control quadrotors with onboard tools (sensors, information processing, control). An early study was published by Nardi and Holland [84] where they discuss a few of the challenges and propose an automated system identification approach to generate an appropriate model of micro aerial vehicles (MAV).

4.6 Collective Manipulation

The rather early study by Ijspeert et al. [192] from 2001 is an evergreen of swarm robotics because they obtained particularly interesting results concerning swarm performance. The investigated task is "stick pulling." Robots are equipped with a gripper and are supposed to pull sticks out of the ground. The experiment is designed in a way, however, that a single robot cannot pull the stick completely out of its hole in the ground. Pulling a stick out, hence, requires two robots and a sequence of the following actions. First, a robot needs to find a stick. Second, the robot has to do the so-called grip1, which moves the stick out of the hole about half-way. Third, the robot has to wait in this position for help. Fourth, a second robot may come by (if the first robot does not just give up, releases the stick into the hole and moves on) and, fifth, does grip2 to remove the stick completely. The third step (waiting for help) comes with a crucial parameter: the waiting time. If the robot waits for too short, it is unlikely that another robot comes by in that short period of time. If the robot, however, waits for too long, the swarm may be quite inefficient or even worse we may run into a deadlock situation because all robots do grip1 and wait.

Another question is that about the optimal swarm size for a given environment and waiting time. Ijspeert et al. [192] observed a superlinear performance increase (cf. Sect. 1.2.1) going from swarm size $N = 2$ to $N = 6$ robots. For swarm sizes $7 \leq N \leq 11$, the performance increase is linear. For swarm size $N > 11$ the performance breaks down because the robots interfere with each other. Hence, we have a nice swarm effect in this scenario. A single robot receives zero performance and for $2 \leq N \leq 6$ the performance increases superlinearly, that is, each swarm member gets more efficient with increasing swarm size. That is a property we usually do not observe in human groups.

4.7 Flocking and Collective Motion

Flocking is the coordinated motion of bird flocks (see Fig. 4.8). At any time, all birds of the flock fly about in the same direction with the same speed. In one of the most classical swarm papers, Reynolds [327] published in 1987 three simple rules that allowed him to simulate a bird flock. The rules require the definition of a neighborhood, for example, by a maximal distance. (1) Alignment: adapt your direction to that of your neighbors. (2) Cohesion: stay close to your neighbors. (3) Separation: if you are too close to one of your neighbors, then avoid a collision. The original concept of Reynolds [327] also required to ignore agents in the back of an agent.

The approach of Reynolds is metric because neighborhood is defined on distances. Meanwhile we know that birds probably use a topological approach, that is, they check, for example, their five closest peers and ignore distances when determining their neighborhood [27]. The accompanying classical theoretical paper for Reynold's work is that of Vicsek et al. [400]. They report a kinetic phase transition, which is interesting from a physics point of view. Another early theoretical work on flocking is that of Toner and Tu [382] and Savkin [339] does control theory for flocking in autonomous mobile robots.

Flocking behaviors can easily be simulated to gather some data that may be of interest. In Fig. 4.9a you see such a measurement from a simulation based

Fig. 4.8 Starlings as a natural example of flocking (Photo by Oronbb, CC BY-SA 3.0)

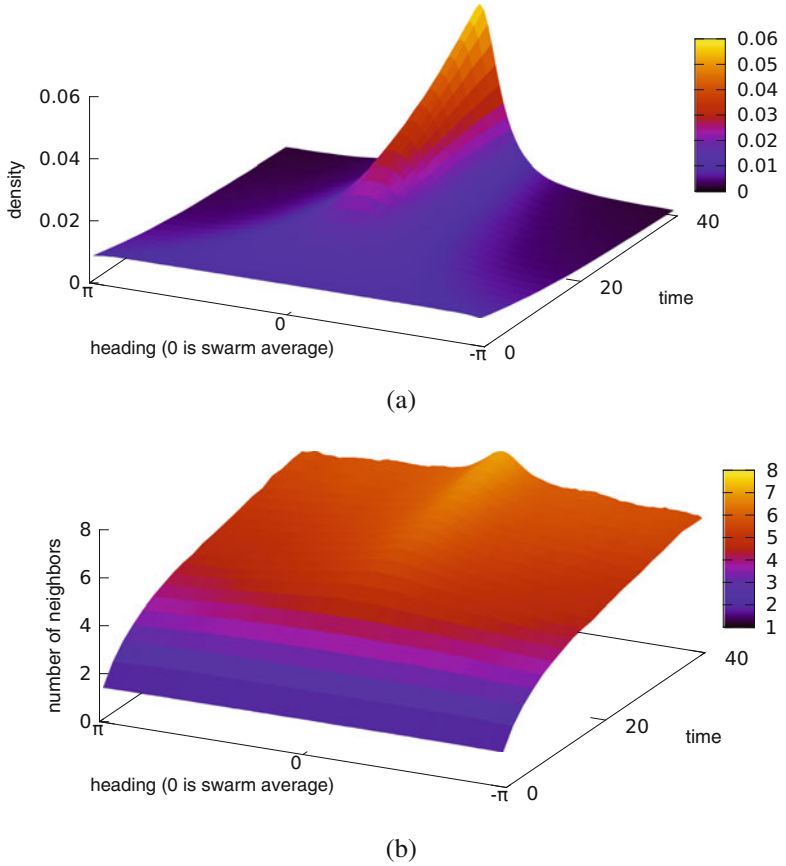


Fig. 4.9 Measurements of probability density of heading and connectivity in a simulated flocking swarm. (**a**) Heading (**b**) Connectivity

on Reynolds [327]. It shows the probability density of all headings in radians (from $-\pi$ to π) found in the swarm at a given time. The headings are actually relative to the current average heading of the swarm. This allows to average data over several independent simulation runs. Initially this density is homogeneous because the headings are assigned uniformly randomly. Over time a peak around heading zero emerges clearly. This is the expected flocking behavior with all agents moving approximately in the same direction. In Fig. 4.9b you see another measurement from the same simulation runs. This one shows the connectivity, that is the number of neighbors, depending on the agents heading. Initially the connectivity is below two. That is an arbitrary value due to the chosen swarm density in this particular experiment. Over time the connectivity increases independent of the agent's heading. Only towards the end of the measurements a peak emerges around heading zero again. However, the difference between the average connectivity of an

agent with heading π (i.e., 180°) and heading zero is small and even less than one. This data shows that flocking also comes with an aggregation behavior probably due to the cohesion rule. At the same time we can say that all agents in this flocking simulation are save and are unlikely to lose connection. Even if an agent temporarily flies in the totally wrong direction of heading π, it still has about seven or eight neighbors that may help it to turn around following the alignment rule.

Collective motion is a diverse field as reflected, for example, by the extensive review paper of Vicsek and Zafeiris [401]. A particular example is the paper by Yates et al. [431]. In this rather theoretical work they investigate the collective motion in locusts. In their nymph state, locusts cannot yet fly. Instead they show collective motion, which is called "marching bands" because they walk in long groups of extremely many animals [59]. Instead of measuring their behavior in the wild, biologists prefer to reduce the complexity and bring them into the lab. There the locusts are allowed to move within a ring-shaped arena. Depending on the swarm size and swarm density one observes a synchronization in their walking directions. From the two options of clockwise motion and counterclockwise motion they choose by majority only one (cf. collective decision-making, see Chap. 6).

It is probably not easy to accept that flocking turns out to be rather difficult to be implemented in real robots. Probably most problematic is the alignment rule because it assumes that each robot knows the heading of all its neighbors. Working with sensors only and without communication, it is unclear how the heading of neighbors can be perceived. While the birds probably follow visual cues, swarm robots usually do not have a good vision system. Even a single camera may not help because of the wide range that needs to be covered. Once we allow communication, it is still unclear what information should be transmitted. If the robot has no global information, it can only try to communicate some information about a relative angle between the two considered robots. If the robots have digital compasses, which is arguably giving them access to global information, then a robot can simply transmit its absolute heading. This is the approach taken by Turgut et al. [388]. They use a digital compass and communicate the heading via a wireless communication module (IEEE 802.15.4/ZigBee compliant XBee wireless module). They report successful experiments with seven robots (they use the so-called Kobot). One of the earliest approaches to group formation in robotics goes back to Yamaguchi et al. [425], however, the distributed control scheme is only validated in simulation.

A different approach is followed by Moeslinger et al. [282]. They propose a simplified flocking algorithm for simple swarm robots that does neither need communication nor global information. The algorithm relies on localization of neighbors according to a sophisticated sector design. There are virtual sectors around each robot and we assume that a robot can distinguish four directions to its neighbors. This can be implemented with at least four infrared sensors. For detecting neighbors in each sector there are rules whether the robot should turn or keep moving. For example, if there is a neighbor close-by, the robot will turn away (separation). If there is a neighbor relatively far at the sides or in the back, the robot turns towards it (alignment and cohesion). Once the sectors are cleverly chosen and the different rules come with the correct priorities flocking is achieved. Moeslinger et al. [281] validated that behavior in a tiny swarm of three e-puck robots.

4.8 Foraging

Foraging behaviors are omnipresent in the animal world because all animals need nutrition. We are interested in foraging strategies of social insects. We skip a sophisticated review of the biological literature here. From the modeling perspective of swarm robotics, for example, the foraging model of Schmickl and Crailsheim [343] for honeybees is interesting and so is also the relevance of noise in the decision-making process of foraging ants as reported by Meyer et al. [274]. The work by Jackson et al. [194] was mentioned above in Sect. 4.3.1 and it is relevant here because it explains how foraging ants navigate their trail network.

Foraging is also of interest in robotics even though our robots do usually not eat. Instead you should interpret foraging as a search and retrieval behavior. There are of course many tasks, such as search and rescue, that are related to foraging and of interest in robotics. All these tasks can be done by a single robot or a robot swarm.

Interesting for modeling in swarm robotics is the model by Lerman and Galstyan [236]. They investigate the effect of interference in a foraging group of robots. Robots search objects, pick them up, and bring them to a certain target area. It is expected that there can be quite some traffic at that target area, which may trigger interference between robots. For example, they find the typical swarm performance curves as described in Chap. 1. Hamann and Wörn [175] report a spatial model of a foraging behavior based on pheromones.

Probably one of the earliest works on foraging in robots was reported by Sugawara and Sno [369] and Sugawara et al. [368]. They report successful experiments with five robots and find that cooperation is effective.

Mayet et al. [259] report a method to emulate pheromones for a foraging behavior similar to ants based on ultraviolet light and glow paint (see Fig. 4.10). They use extended e-puck robots. Robots can "draw" trails on the ground by lighting their ultraviolet LED. The trail is then detected by vision. The approach also includes a light beacon to emulate a compass.

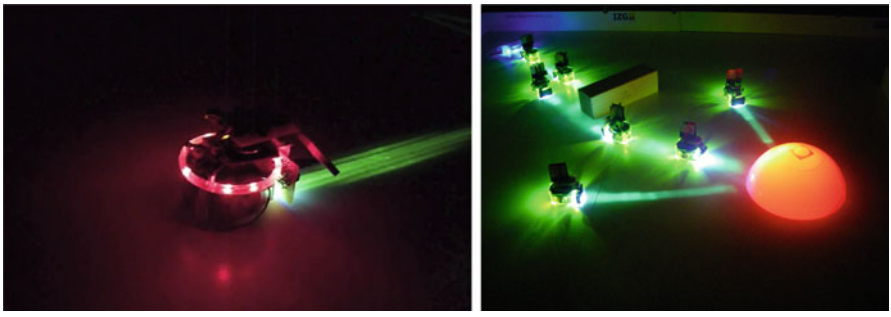


Fig. 4.10 Emulation of pheromones with ultraviolet light and glow paint by Mayet et al. [259]. Note that the right-hand photo is contrived

The above mentioned approach by Nouyan et al. [295] is one of the most sophisticated experiments in swarm robotics. The robots form chains, search for an object, once the object is found they collectively transport it along the chain of robots back to their base station.

The probably most complex foraging behavior reported in swarm robotics is that of Dorigo et al. [98]. It was done within the Swarmanoid project that investigated heterogeneous swarm robots. The eye-bot (quadrotor), hand-bot (robot with grippers), and foot-bot (similar to the s-bot) cooperate to pick up a book from a shelve. First they have to find it, which is done by the eye-bot, then they form a relay-station line of foot-bots, collectively transport a hand-bot to the shelf, and the hand-bot finally picks up the book.

4.9 Division of Labor and Task Partitioning/Allocation/Switching

The challenge of properly partitioning and allocating work in a swarm is important, difficult, and is an aspect of swarm robotics that comes with a lot of algorithmic features. Division of labor means the work is organized, not everyone is doing the same tasks, and consequently the work needs to be usefully distributed among the workers. Task partitioning is the problem of defining smaller pieces of work (tasks). They should of course be self-contained and each task should have similar requirements and a similar volume. Task allocation is then the problem of assigning tasks to the robots, hopefully in an efficient way. Both task partitioning and task allocation can be done offline (before deploying the system) and online (at runtime). An offline approach is simpler but an online approach allows for adaptations to dynamic changes. If the task allocation is dynamic, then a strategy for task switching is required. One wants to avoid frequent switching because each switch may come with a certain overhead.

Again it is useful to look at biological role models, in particular social insects. Ratnieks and Anderson [323] define division of labor as

> the division of the workforce among the range of tasks performed in the colony, whereas task partitioning is the division of a discrete task among workers.

In their paper Ratnieks and Anderson [323] review task partitioning in insect societies with examples in ants, bees, wasps, and termites. They say benefits of task partitioning

> occur either through enhancement of individual performance or through enhancement of the overall system (e.g., where partitioning itself eliminates a constraint affecting task performance, such as when a forager can collect sufficient material for several builders).

Anderson et al. [8] study task partitioning in insect societies using the example of bucket brigades.

Gordon [149] reports early studies on task allocation in social insects. She identifies internal factors (e.g., body size) and external factors (e.g., amount of already collected nectar, number of workers engaged in the task) for task allocation. Necessary conditions for task allocation are "local cues that reflect the numbers of workers performing a task and the extend to which the task is accomplished" [149].

An interesting model of task partitioning is reported by Karsai and Schmickl [208]. The question of how to obtain knowledge about the progress of a task based on local cues as described by Gordon [149] is answered here by the concept of a "common stomach." Karsai and Schmickl [208] investigate nest construction in social wasps. The common stomach is formed by a group of wasps who are generalists and can store water in their crops. The common stomach then serves not only as a storage of water but also as an information center about how well the task partitioning is done currently. A set of positive and negative feedbacks is based on the common stomach and regulates the task partitioning process.

The well-known task allocation in honeybees determined by their physiological age is taken as inspiration by Zahadat et al. [433], Zahadat and Schmickl [435]. Their focus is on regulating robot's physiological age based on their local information about the current recruitment demand of their task. The approach is tested in an AUV swarm (autonomous underwater vehicles) with the task of monitoring at different depths.

Enough about the biological systems, now we turn to the multi-robot and robot swarm systems. Nair et al. [292] investigate task allocation in the RoboCup Rescue Simulation Domain. The approach is distributed and based on an auction system. All-to-all communication is required and hence the system does not scale.

Lemaire et al. [232] discuss task allocation in a multi-UAV system (unmanned aerial vehicle system). The architecture is completely distributed and tasks are also allocated in an auction process. Centralized planning is avoided but the system is synchronized by token circulation and all-to-all communication is required. The approach is verified in simulation only.

Kanakia [206] presents an approach to task allocation based on a response threshold approach. The focus is on allowing only local information and local communication. A theoretical result is a proof that the system converges. Kanakia [206] makes use of game theory, in particular so-called global games: "Global games can be seen as a particular instance of equilibrium selection through perturbations." [64]. The idea of global games is to focus on "strategic environments with incomplete information." Global games come with an essential question of committing or not committing. An interesting example is a bank run (see Fig. 4.11), the question of withdrawing your money from the bank in a crisis or not [146]. Kanakia [206] investigates scenarios similar to the stick-pulling task allocation problem (see Sect. 4.6), macroscopic modeling includes a master equation approach, and also verifies the results in physical experiments using the Droplet swarm robot (see Farrow et al. [116] and Sect. 2.5.5).

An approach to task allocation in sequentially interdependent tasks is presented by Brutschy et al. [58]. The robots measure the delay between having finished their task and having to wait for another subtask to finish. They use this local cue to decide about task switching. The approach relies on these measured delays and on a probabilistic task switching behavior. The results are verified in simulation.

Fig. 4.11 Example of a bank run, group of people in front of the closed American Union Bank, New York City, April 26, 1932 (public domain)

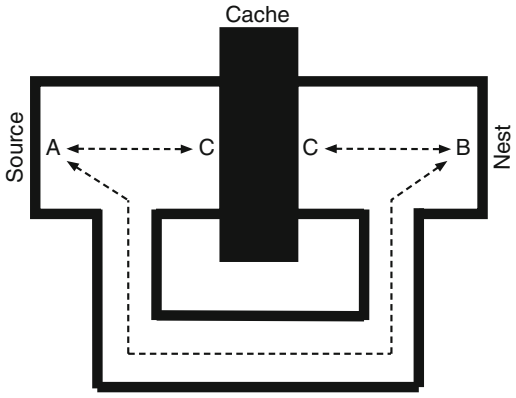


Fig. 4.12 Task partitioning scenario as described by Pini et al. [309]. Robots can decide against task partitioning and transport objects from *A* to *B* directly, or a robot can decide to go for task partitioning and transport objects only from *A* to *C* or from *C* to *B*

Pini et al. [309] present an approach for task partitioning that is based on methods from machine learning. They consider task partitioning, in particular the question of deciding at runtime whether to make use of task partitioning at all. They apply well-known methods from the multi-armed bandit problem. The considered swarm has to transport objects from *A* to *B* and robots can choose to use a cache of objects *C* (see Fig. 4.12). Hence, a robot can do the full unpartitioned service of transporting objects from *A* to *B*, or a robot can decide to go for task partitioning and transport objects only from *A* to *C* or from *C* to *B*. The robots do not know the costs of either of these three options beforehand. Hence, they have to explore and estimate

the costs online while working on the task. This corresponds directly to the typical challenge in the multi-armed bandit problem. A gambler can choose to play any of the bandits' arms but initially doesn't know which one gives in average the biggest profit. The gambler, or in our problem the robot, has to explore the different options (exploration), estimate the underlying probability density, and make a choice of which option to go for (exploitation). In addition, the costs or the payoff may be dynamic, that is, each option needs to be explored from time to time again. Pini et al. [309] directly apply methods from machine learning to task partitioning in swarm robotics. There is no robot-to-robot communication and they verify their approach in simulation.

Khaluf et al. [215] present a specific approach to task allocation and task switching with soft deadlines in swarm robotics. A soft deadline is a deadline that should be met but if it is not met the quality of the result is only decreased in quality but has no catastrophic results. The approach is probabilistic and offline, that is, each robot is equipped with a so-called decision matrix that defines probabilities of switching between tasks at runtime. This matrix is not changed at runtime but pre-calculated and optimized offline. The system is still adaptive to changes in the swarm size, for example, in the case of robot failures. There is no robot-to-robot communication and results are verified in simulation.

Ferrante et al. [119] use evolutionary algorithms in a robot swarm to evolve behaviors that show specialization, that is, task partitioning. They investigate a scenario inspired by leaf-cutting ants that has potential for task partitioning. The ants can divide the work into two jobs: (1) dropping leaves from trees and (2) transporting them to the nest. Neither the basic behaviors of what to do when nor the actual task partitioning is predetermined. Still, Ferrante et al. [119] observed a self-organized, emergent task partitioning in the system.

4.10 Shepherding

Shepherding behaviors are interesting for swarm robotics because they often require more than one shepherd robot, the behaviors of the shepherds are mutually dependent, and the multi-robot approach is not a mere parallelization of a task that could be done also by a single robot. There is an early work by Schultz et al. [351] using genetic algorithms to learn behaviors for shepherding. A similar approach is reported by Potter et al. [314], which focuses on a heterogeneous approach and the emergence of specialists. One step further but following a similar line is the study of Gomes et al. [147]. They investigate a coevolutionary approach with a special technique of merging and splitting populations.

Hardware implementations of shepherding robot swarms seem not to exist yet. There is "SwagBot" that was developed at the Australian Centre for Field Robotics, University of Sydney, which can herd cattle.[1]

4.11 Heterogeneous Swarms

In Chap. 1 we have defined robot swarms as (quasi-)homogeneous systems following Beni [37]. However, in recent works advantages of using heterogeneous robot swarms have been investigated. For example, Arkin and Egerstedt [14] discuss heterogeneity in robotics in general and point to a niche that seems not well covered yet and that they nickname "slowbots." Just as in the animal world we see animals across all scales from super-fast to super-slow, we might want to think about a similar diversity in robotics. The above-mentioned project Swarmanoid by Dorigo et al. [98] explicitly investigated heterogeneous hardware in swarm robotics with their eye-bots, hand-bots, and foot-bots. Especially the combination of flying robots and ground-based robots seems an advantageous approach, for example, by using the visual overview of a quadrotor to instruct wheeled robots on the ground. This way still local information is used and the system can scale up but each robot has much more information available. Kengyel et al. [209] investigate the impact of software heterogeneity (i.e., the robot hardware is homogeneous but robots have different controllers) in an aggregation task inspired by a honeybee behavior. They use methods of evolutionary algorithms to combine appropriate numbers of predefined robot types. Prorok et al. [319] investigate the impact of diversity in a robot swarm on performance in a task allocation problem, where each task has different requirements for the robots working on it. They also define a diversity metric called "eigenspecies" to quantify differences in heterogeneity. In a much earlier work Balch [25] defined a measure of heterogeneity called "hierarchic social entropy." It takes into account what fraction of the swarm is of the same kind and the differences between the types. Another measure of heterogeneity by Twu et al. [390] stresses that heterogeneity basically depends on two components: complexity and disparity. Complexity describes the overall number of differences. For example, if not two robots are of the same kind, then the complexity is high. Disparity describes how much the overall difference between robots is maximized. For example, if there are only two groups of robots but both are at one opposing extreme end of the "diversity scale," then the disparity is high. For further reading, see [105, 178, 245, 258].

[1] https://spectrum.ieee.org/automaton/robotics/industrial-robots/swagbot-to-herd-cattle-on-australian-ranches.

4.12 Mixed Societies and Bio-Hybrid Systems

Mixed societies and bio-hybrid systems in the context of swarm robotics are systems of interacting robots and living organisms. Although it may seem scary and it may remind oneself of cyborgs from the science-fiction literature, they may actually prove to be useful in the future. If robots, for example, are used to control an insect swarm, this opens up completely new options for pest control. Also a symbiosis between robots and organisms is possible to get the best out of both worlds combined in one system.

The probably most prominent example is the work by Caprari et al. [63] and Halloy et al. [163] on a mixed society of cockroaches and robots. In the project called "Leurre" they developed a small, simple mobile robot called "Insbot" that was able to usefully interact with cockroaches (see Fig. 4.13). Although one may think that it must be hard for such a robot to be accepted by the cockroaches as peer, Halloy et al. [163] succeeded in doing so. It seems to require not much, as the robot was just clad in a sort of blotting paper that was soaked with cockroach odors. The task for the robots within this mixed society was to socially influence the cockroaches. The society was put in an arena with two shelters of different size. Under normal circumstances the cockroaches would take the shelter that allows all cockroaches to just fit beneath it. However, the robots were programmed to try to convince the cockroaches that aggregation beneath the other shelter is more desirable and they succeeded. A similar study was done by da Silva Guerra et al. [82] with robots and crickets.

In the "Chicken Robot" project Gribovskiy et al. [152] developed a robot that is able to interact with young chicken (see Fig. 4.14). The robot makes use of the relatively popular, so-called filial imprinting: "shortly after hatching we present the moving and vocalizing robot to chicks and they learn that the robot is the mother" [152].

Another research project that was fully focused on bio-hybrid systems was ASSISI$|_{\text{bf}}$ (see Fig. 4.15). The researchers investigated two different setups, one with multiple stationary robots and honeybees [249] and another one with a mobile robot and zebra fish [51]. The idea is to establish a form of communication between the robots and the animals (hence the project's name). Also the robots are supposed to influence the animals in possibly user-defined ways.

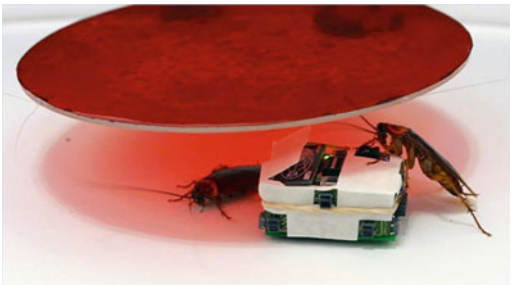


Fig. 4.13 A bio-hybrid system of cockroaches and an "Insbot" robot [357]

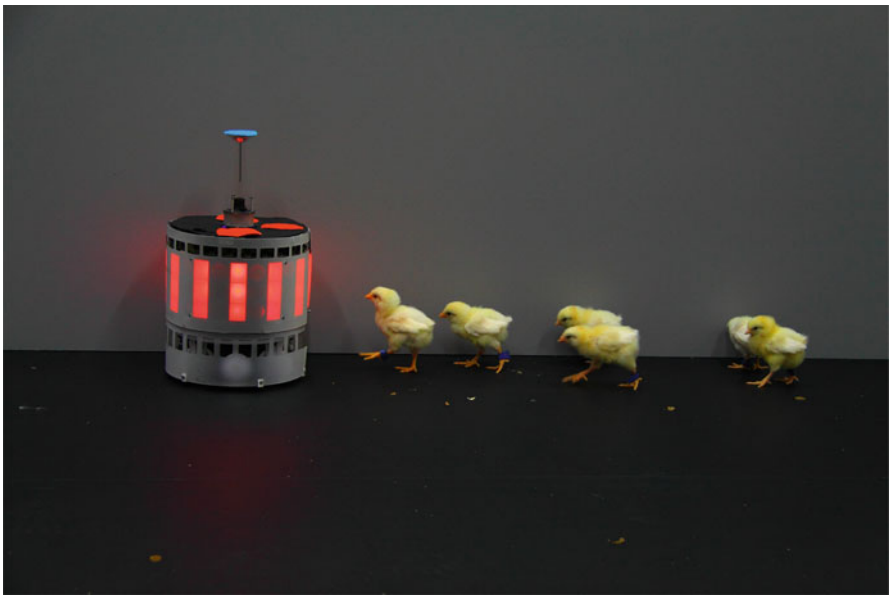


Fig. 4.14 "Chicken Robot" project Gribovskiy et al. [152] (photo by José Halloy, with permission)

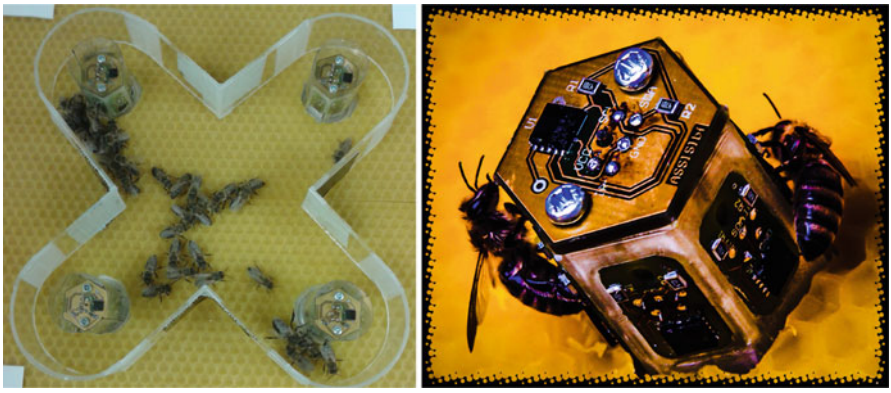


Fig. 4.15 A bio-hybrid system of interacting robots and honeybees in the project ASSISI|$_{bf}$. The robots were developed by FER, University of Zagreb (photos by Thomas Schmickl, Artificial Life Lab, Karl-Franzens University of Graz)

In the project *flora robotica* a bio-hybrid system of robots and natural plants is formed [121, 167, 174]. The robots are stationary and control the directional growth of the plants by blue light of bright LEDs (see Fig. 4.16). The idea is to use this concept in applications of architecture. For example, the bio-hybrid *flora robotica* system could autonomously grow green walls, roofs, or maybe in the future even houses.

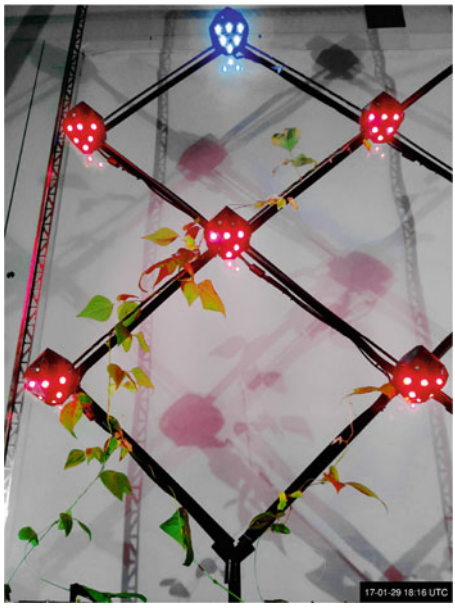


Fig. 4.16 The bio-hybrid *flora robotica* system of robots and natural plants by Hamann et al. [167]

4.13 Swarm Robotics 2.0

There exists a kind of "second wave" of an effort for swarm robotics. Swarm robotics research needs to find its way out of the lab into real-world applications. The transition from swarm robotics as an indoor and lab-based research to swarm robotics as field robotics is feasible and starting just now. Other aspects of swarm robotics that were neglected or even ignored before include questions of robustness, such as error detection and general security issues, and also psychological effects on humans (see Sect. 1.6). Other novel concepts for swarm robotics include the use as haptic interface/display by Kim and Follmer [217] (see Sect. 4.13.2) and the approach of "robotic material" by McEvoy and Correll [260].

4.13.1 Error Detection and Security

Robot swarms are already robust because of the high degree of redundancy. However, on closer look redundancy is rather only a potential for robustness. Robustness requires more than just replacing robots that fail completely. What happens, for example, if a robot does not just break down but actually causes problems because it is still partially operational. Such a robot could cause errors and damage because other, fully functional robots may react in funny ways to its faulty behavior. In a hypothetical scenario a faulty robot could deposit a large amount of pheromone at one spot attracting many robots to an irrelevant position. Hence, it is

necessary to detect errors and to ignore faulty robots. Concerning security, we can also imagine a situation when a saboteur has a robot infiltrate a swarm. The robot may manage to be accepted as peer and may start to behave strangely.

Christensen et al. [71] address the problem of detecting faulty behavior by making all robots synchronize following the firefly algorithm. Robots are supposed to flash in synchrony. In their "fault detection scheme, the periodic flashes function as a heart-beat mechanism" [71]. A faulty robot can be detected if it does not synchronize properly. They show in robot experiments that "robots are able to detect and respond to faults by detecting non-flashing robots" [71]. The swarm stays robust even if there are multiple faults and the robot swarm has self-repair capabilities to survive a relatively high failure rate.

A more sophisticated approach is reported by Tarapore et al. [374]. The approach is inspired by the immune system [65] and has a crossregulation model [375]. The crossregulation model has to learn online what a normal and what an abnormal behavior is. Robots use feature vectors to describe observed behaviors and share them with their neighbors. Finally, robots have a vote to consolidate their models and to determine faulty robots.

In a rather speculative paper, Ferrer [120] discusses options of using a blockchain approach for robot swarms. The decentralized approach based on peer-to-peer networks can be transferred to swarm robotics and, for example, intruders may be detected by analyzing the blockchain. However, it is unclear how the resource intensive requirements (computational power and memory) can be satisfied with typical robot-swarm hardware or how these requirements can be reduced. An alternative to blockchains could be any standard technique to encrypt and sign messages. The added value of blockchains compared to their required resources is rather questionable.

4.13.2 Interfacing Robots and Robots as Interface

A novel idea for swarm robotics that relates to the concept of "robotic material" [260] are "ubiquitous robotic interfaces" [217]. The robot swarm displays information to users (e.g., collective motion to point to objects or to operate as timer) but can also manipulate (e.g., pushing objects around) and sense (e.g., their location and surrounding objects). A new aspect is user interaction that allows the user to move around a robot and using it as an input device. Kim and Follmer [217] showcase their approach with a swarm of ten small differential-drive robots (diameter 26 mm). The swarm is able to perform standard behaviors, such as aggregation, dispersion, circling, random motion, and flocking. Localization is implemented as a central system based on a DLP projector that produces a gray code pattern. Robot swarms as robotic interface may have a future but they need to be decentralized, probably even smaller, and reliable to be of use.

The more common interface aspect is the question of how to create an interface that allows, for example, a human supervisor to efficiently interact with the robot

swarm (another flavor of the human factor, cf. Sect. 1.6). There is a number of recent publications on that indicating that a new field of human–swarm interaction just emerged and gets a lot of attention.

An overview is given in a survey paper by Kolling et al. [219]. They discuss essential aspects, such as remote interaction with the swarm, proximal interaction (supervisor is with the swarm and the robots can perceive the supervisor), visualization, control, levels of automation, and organization in sub-swarms. Most of that goes beyond the scope of this book but a few example papers should be mentioned. Different ways of proximal interaction are discussed by a number of authors, such as using gestures to control a robot swarm by Nagi et al. [291] and Monajjemi et al. [284] for unmanned aerial vehicles (UAV). Mondada et al. [287] propose to use electroencephalography (EEG) signals, that is, basically preparing the sci-fi dream of brain controlled robot swarms. Pourmehr et al. [315] use face engagement and voice commands to control groups of robots. Harriott et al. [179] define human–swarm interaction metrics, for example, the important distinction between micro-level movements and macro-level movements.

4.13.3 Swarm Robotics as Field Robotics

Duarte et al. [103] discuss an important issue about swarm robotics (also see Duarte et al. [104]):

> Despite the potential of swarm robotics for real-world tasks, no demonstrations of swarm behaviors outside of controlled laboratory conditions have been performed so far [...] all real-robot experiments have, in fact, been performed in controlled laboratory environments.

Duarte et al. [103] themselves report their first exception of that rule: a swarm of ten aquatic surface robots. The robot is a differential drive mono-hull boat. The onboard control runs on a Raspberry Pi 2. The robots have IEEE 802.11g WiFi, GPS, and a compass. Reported experiments include homing (navigation to waypoints), dispersion, aggregation, and area monitoring.

Probably it is not mere coincidence that also the second outdoor swarm robot experiment is done in water (cf. the quote of Frank Schätzing at the beginning of this chapter; maybe we still underestimate the importance of our oceans). Within the European-funded project "subCULTron"[2] researchers develop an underwater swarm (culture) of robots (autonomous underwater vehicles, AUV). The final demonstration will be done in the lagoon of Venice with a heterogeneous robot swarm of three types of robots and many dozens of robots in total. The main task is monitoring. For some early impressions, see Fig. 4.17.

Another example is the "Swarm Robotics for Agricultural Applications" (SAGA) project by Trianni et al. [385]. Also this project refers to the get-out-of-the-

[2] http://www.subcultron.eu/.

Fig. 4.17 Work within the project "subCULTron" at the lagoon of Venice. The robots were developed by FER, University of Zagreb and SSSA Scuola Superiore Sant'Anna (photos by Thomas Schmickl, Artificial Life Lab, Karl-Franzens University of Graz)

lab meme: "swarm robotics research is still confined into the lab, and no application in the field is currently available." Also SAGA does not investigate robots on land but up in the air. A group of quadrotors is supposed to help in controlling weed, such as detecting weeds and mapping infested areas. The quadrotors fly on relatively low altitude to inspect plants on the ground visually. The robots use GPS and onboard vision. One main challenge is to do efficient task allocation online.

These applications of swarm robotics are in water and in the air and not on land probably because reliable robot control in water/air is easier than on land. Wheeled or legged robots can get stuck, fall over, and damage themselves or obstacles. Swarm robotics faces there the same challenges as robotics in general. In order to create a reliable robot one usually needs a lot of sensors (cameras, laser scanners, radar) and computing power. Complex sensors are too big, too heavy, and too expensive for swarm robotics. Basically it means that we are still not yet able to build any robot that gets even only remotely close to the capabilities of an insect. However, if insects can control their legs and navigate robustly, then also engineers will be able to achieve the same on similar scales one day. We should accept the challenge.

4.14 Further Reading

The review papers by Brambilla et al. [54] and Bayindir [32] basically go through different scenarios of swarm robotics. Floreano and Mattiussi [122] give a short overview of collective systems including reconfigurable robotics and co-evolutionary systems.

4.15 Tasks

4.15.1 Task: Behaviors of Robot Swarms

Initialize a robot arena with 20 randomly distributed robots. All robots should be operated by instances of the same controller (i.e., copies of the same controllers).

(a) Program a behavior that makes robots stop if they get close to another robot. For that your simulated robots require to have sensors that detect close-by robots. Possibly you can also fake such a sensor based on distances between robots to keep the simulation simple.
(b) Extend your program to limit the time a robot stays stopped according to a defined waiting time. When a robot wakes up and moves again it might immediately stop again depending on your implementation. However, the idea is that the robot leaves at least small clusters before stopping again. Think of a strategy to implement this behavior and change your program accordingly.
(c) Tweak your implemented behavior and parameters such that your robot swarm eventually aggregates in one big cluster with robots leaving only from time to time but rejoining it soon again.

Chapter 5
Modeling Swarm Systems and Formal Design Methods

That relatively simple algorithm is how my model works, and it manifests itself as complex hunting behavior when scaled up to a swarm of stigmergic agents.
—Daniel Suarez, Kill Decision

Leon and I could try to model a swarm of them electronically. We could give them various characteristics and see how long it takes for them to start acting like a brain.
—Frank Schätzing, The Swarm

Abstract We learn about why and how we model systems of swarm robotics and how sophisticated design methods can look like.

Modeling is motivated and introduced as a dimension reduction technique for swarm robotics. Then we start from a discussion of local sampling, which is the challenge in swarm robotics of dealing with local information. The local samples are not representative for the whole swarm and hence we require a methodology to work with unreliable local information. Several modeling techniques are introduced that are frequently applied in swarm robotics, such as rate equations and spatial models based on ordinary and partial differential equations. We also discuss network models and the interesting option of using robot swarms as models for biology.

In the second part, we turn to formal design approaches starting from multi-scale modeling. Software engineering approaches and verification techniques are discussed and we look into the so-called concept of global-to-local programming.

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2_5

5.1 Introduction to Modeling

5.1.1 What Is Modeling?

Modeling is an essential part of the scientific approach in general. Modeling of systems has usually three objectives: abstraction, simplification, and formalization. Abstraction means to intentionally omit parts of the modeled system. Simplification means to transfer the modeled system into a simpler system that is easier to understand. Formalization means to describe the modeled system by a formal system that allows to derive new insights by means of logic and mathematics. The option of getting new insights about an investigated system by means of formal derivation is important because this is how arguably half of scientific progress is made (cf. history of science, such as Paul Dirac's prediction of neutrinos, etc.). The other half is made by a purely creative approach of gaining a deep (possibly informal) understanding of the system first (cf. history of science, such as Werner Heisenberg's creation of matrix mechanics, etc.).

Why is it important for swarm robotics to model swarm systems? The problem is that swarm robotics systems consist of many entities and hence are of high dimension. For example, consider a swarm of size N in two-dimensional space. Then each agent $i \in \{1, 2, \ldots, N\}$ has a position $\mathbf{x}_i \in \mathbb{R}^2$ with $\mathbf{x}_i = (x_i^1, x_i^2)$, a velocity $\mathbf{v}_i \in \mathbb{R}^2$ with $\mathbf{v}_i = (v_i^1, v_i^2)$ which includes the agents direction, and a discrete state $s_i \in \mathbb{N}_0$. Now we can define any of the system's configurations as

$$\gamma = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N, s_1, s_2, \ldots, s_N), \tag{5.1}$$

which defines also a configuration space Γ with $\gamma \in \Gamma$. This space has dimension $\dim(\Gamma) = 2N + 2N + N = 5N$. For a big robot swarm of size $N = 1000$ we get a huge 5000-dimensional space. There is no good way of trying to understand how such a huge system works except for running simulations. The simulation actually keeps track of all $5N$ dimensions, updates each variable in each time step (in a simulation we have discrete time), and maybe even visualizes the system. That way we can observe sequences of configurations (or trajectories) $\gamma_t, \gamma_{t+1}, \gamma_{t+2}, \ldots$ However, we can only observe one sequence at a time for a specific initialization γ_0. What we ideally need instead is to understand how the system operates in general for any setup. Therefore, it is obvious that we require to reduce the high-dimensionality of the system by a modeling approach, that is, by abstraction and simplification. We need to omit certain parts of the system such that we obtain a different definition of a configuration, say φ, with a smaller configuration space $\varphi \in \Phi$. We would like to have $\dim(\Phi) \ll \dim(\Gamma)$. For example, we could try to find a mapping f that maps the actual configurations γ to the reduced configurations φ:

$$f : \Gamma \mapsto \Phi. \tag{5.2}$$

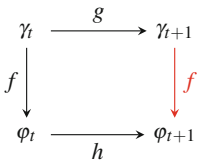


Fig. 5.1 Modeling represented as a series of mappings. The requirement given in Eq. (5.3) should hold

The map f implements an abstraction which is already a part of our modeling approach.

We consider for now a discrete time model. Then we can define two update rules as maps $g : \Gamma \mapsto \Gamma$ for the real system and $h : \Phi \mapsto \Phi$ for the model. They determine how the system develops in time. For a given configuration γ_t at time step t we determine the configuration of the next time step $t + 1$ by $g(\gamma_t) = \gamma_{t+1}$ and similarly for h. Now we can put everything together as shown in Fig. 5.1. The evolution in time of the real system is directly determined by the map g. If we are provided with a configuration γ_t and we want to know the configuration that our model predicts, then we need first to map γ_t to the model space Φ using f to obtain φ_t and then we can apply the update rule h to obtain φ_{t+1}. We can now compare the configurations γ_{t+1} and φ_{t+1} by applying f to γ_{t+1} again. We would like to have

$$h(f(\gamma_t)) \stackrel{!}{=} f(g(\gamma_t)), \tag{5.3}$$

that is, the modeling abstractions implemented by f should be chosen carefully such that the model update rule h is able to predict the correct model configuration for the next time step. Also note that we could try to define an inverse map $f^{-1} : \Phi \mapsto \Gamma$ that reverses the model abstractions and rebuilds the configuration γ of the real system from the model configuration φ. However, typically f^{-1} cannot usefully be defined because f is surjective, that is, we can have $\gamma_1 \in \Gamma$ and $\gamma_2 \in \Gamma$ with $\gamma_1 \neq \gamma_2$ and $f(\gamma_1) = f(\gamma_2)$. This is also easily accepted by thinking of a possibly extreme abstraction f that reduces a high-dimensional system to a low-dimensional model. Obviously we would need to throw away a lot of details to achieve that. Once we try to reverse that abstraction we would need to add all kinds of details without any information about which details were omitted before. For the inverse map f^{-1} that means that it can in general only define an arbitrary representative γ for a model configuration φ.

5.1.2 Why Do We Need Models in Swarm Robotics?

The main objective of modeling in swarm robotics is dimension reduction. We need to abstract away microscopic details in order to detect key features and general principles of operation in the system. This is also nicely stated by Schweitzer [354]:

> To gain insight into the interplay between microscopic interactions and macroscopic features, it is important to find a level of description that, on the one hand, considers specific features of the system and is suitable for reflecting the origination of new qualities, but, on the other hand, is not flooded with microscopic details. [...] A commonly accepted theory of agent systems that also allows analytical investigations is, however, still pending because of the diversity of the various models invented for particular applications.

He is also pointing to the problem that we do not have yet a common model approach that can be applied to any scenario. Instead many different models have been developed and have been applied to different setups of swarm robotics systems.

The abstraction of microscopic details that fights the "microscopic flood" could in principle also be done automatically in a brute-force approach with methods from machine learning and statistics. Options would be the principal component analysis or feature extraction. However, experience shows that these approaches are not constructive. They reduce the dimensionality but the obtained reduced configuration space Φ would have in general no explanatory power and no meaning.

What we need is a reduced configuration space Φ that still contains and can represent key features of the modeled system. What a key feature is depends on the respective application. In a collective decision-making system, for example, the current status of who is in favor of which option is certainly a key feature but the agent positions might be of less relevance. In an aggregation scenario the distance between agents is a key feature but maybe their internal state is of less importance. In general we choose whether we want to model time and/or space. If we choose to represent the temporal course of the system, that is, we have a time-dependent model, then we can choose between a discrete- or continuous-time model. If we choose to represent the position of agents or agent densities, that is we have a spatial model, then we can also choose between a discrete or continuous representation of space. It turns out that many of our models cannot be solved analytically, which means we have to solve them numerically using computers. If you are a computer scientist, then you might want to take a practical position in saying that in this case we should just always think of discrete models. However, the notation of a continuous model is often concise and easy to handle. It also allows for a simple comparison between models, which is important to quickly determine whether two proposed models are different and how they differ.

Let us consider a simple example using the above developed notation. We consider a binary collective decision-making scenario. Binary means that there are only two options, say, A and B. Furthermore, we assume it is a two-dimensional spatial system and each agent $i \in \{1, 2, \ldots, N\}$ has an internal state representing its current opinion $s_i \in \{A, B\}$. The dimension of the configuration space Γ is $\dim(\Gamma) = 2N + 2N + N = 5N$. We define an extreme model abstraction via map

$$f(\gamma) = \frac{|\{s_i | s_i = A\}|}{N} = \frac{[\text{number of agents in favor of option } A]}{N} = \varphi. \tag{5.4}$$

Here φ represents the key feature, which is the decision-making process. The dimension of the model is $\dim(\Phi) = 1$. We have reduced the $5N$-dimensional

system to just one dimension. The important question to ask is now: how does the update rule ($h(\varphi_t) = \varphi_{t+1}$) of the model look like? This is not a trivial question and will be discussed later.

5.2 Local Sampling

When designing and programming controllers for swarm robotics we face a common challenge because of the lack of information. In many situations it would be helpful to know about the global picture, for example, the size of the swarm or where the currently biggest cluster of robots is positioned. However, an individual robot does only have local perception and can only collect a bit more information by communicating to its neighbors. It turns out that also agents of much higher capabilities face similar challenges in certain situations. For example, everyone who was ever lost in a foreign city knows the problem of only knowing the names of maybe two adjacent streets but without the possession of a city map that information is of limited use. In fact, on a bigger scale the human society had a similar problem in the Sixteenth century when trying to localize the earth in relation to surrounding stars and planets. An example of a regularly recurring problem is to predict the result of votes. Surveying all potential voters would be too expensive therefore only a small subset is checked. How to choose this subset usefully is a scientific question. A similar challenge occurs in market research where, for example, the number of customers for a certain product needs to be predicted, of course, also without asking every potential customer.

Inferring the "global picture" in the context of localization is a standard problem in robotics. A common method is SLAM (simultaneous localization and mapping) which generates a global map of the environment while the robot is trying to localize itself within it. Another example is the important task in RoboCup soccer to localize the ball. The robot team cooperates by sharing each robot's local perception. These local sections of the global picture are then merged to produce a good estimation of the ball's actual position (multi-robot sensor fusion). In RoboCup this is typically done by broadcasts, that is, global communication which doesn't scale.

In swarm robotics the only option is to do local sampling and information can only be shared with neighbors but not with everyone in the swarm. In fact, often a robot relies on the information it is able to gather on itself without acquiring more information by communication. An example is stigmergy, for example, cue-based navigation by pheromones. An agent only knows the local pheromone concentration which defines a direction and carries the information of where to go. Another example is the alignment rule in flocking where a robot checks the direction of its neighbors and aligns to them. The underlying assumption of this process is that the local sample of neighbors gives a good estimate of the swarm's dominant direction of motion. Why it is useful to assume this and why it does not always end in a disaster of a dissolving swarm is nontrivial.

One key ingredient that supports the effectiveness of these local sampling processes is self-organization. An example is the emergence of order in pedestrian flows. Although most of the pedestrians are most likely occupied with other thoughts than how to organize the flow efficiently, often two streams of pedestrians with the same walking direction form. The local sampling here is the detection of the walking direction of the person in front of you. If the person is walking in your direction you follow and if the person is approaching you then you step to the side. The formation of ordered flows based on these simple, local rules is an emergent effect.

5.2.1 Sampling in Statistics

Sampling is of course a process that is well-defined in statistics. First, we define the statistical population which is the set of potential measurements. A sampling is the selection of a subset of elements from such a statistical population. Different types of sampling are known. In a nonprobability sampling some elements of the population have no chance of getting selected. In cluster sampling, elements are selected in groups. For example, all households in a certain street are selected or a certain class of students from some school. In accidental sampling elements are selected that are "close at hand." For example, when a survey is done in a street then the persons that the surveyor accidentally runs into are not particularly selected but they are just close at hand. Obviously it is useful to define a metric that defines the quality of a sampling. The most important quality of a sampling is representativeness. If the selected subpopulation has "similar properties" as the full population, then the sample is representative. The definition of what "similar" should mean here is crucial, nontrivial, and depends on what is of relevance in a particular situation.

As an example for how tricky it is to pick a good subpopulation and for how important representativeness is we take the Literary Digest Disaster. The idea was to do a straw poll for the US presidential election in 1936 of Alf Landon against Franklin D. Roosevelt. The huge number of ten million people was polled by mail and phone and 2.4 million answered. 60% of them were in favor of Alf Landon and the prediction was that he would gather 370 of 531 electors (remember the uncommon voting system of the USA which is based on electors). The actual result, as the educated reader knows, was however very different. It turns out that Landon won only two states (Vermont and Maine, that is only eight electors). The representativeness of the sampling was of minor quality due to a number of facts. The sampling was nonprobability sampling because the sample was generated from lists of registered car and phone owners as well as subscribers of the magazine (Literary Digest). These were mostly well-off households which were typical republican (Landon). In addition mostly voters answered the poll that showed an above-average interest to the vote. Unfortunately anti-Roosevelt voters felt more strongly about the election than Roosevelt voters who subsequently tended to not answer the poll.

5.2.2 *Sampling in Swarms*

Because the sample in swarm robotics has to be local it is a nonprobability and accidental sampling. Swarm members that are too far away cannot be selected and the neighbors are selected because they are "close at hand." Hence the sample's representativeness is at least questionable. There are a number of difficulties. The robot might be part of a cluster of robots that is not representable for the swarm. The robot's state and the states of its neighbors might not be independent, that is, they might be correlated. This might also be supported by the robots behavior that might actually implement this correlation directly. Finally, the swarm density might be too low and therefore the sample might be very small which introduces a strong bias.

We start with a first simple example of local sampling in swarm robotics similar to the scenario we discussed in Chap. 3. Say, in a robot swarm robots can have one of two states, black and white, and robots have to estimate the global fraction of black and white robots currently in the swarm. A robot has only a limited sensor range, which defines its neighborhood. See Fig. 5.2 for an example. In total there are 13 black robots (52%) and 12 white robots (48%). The considered robot (marked with a dot) on the top left can only locally sample. It detects four black robots (66.6%) and two white robots (33.3%, including itself). Obviously the robot's measured estimate differs from the true situation. However, here the robot is relatively lucky and receives a good estimate. In other areas of the arena robots may end up with worse estimates.

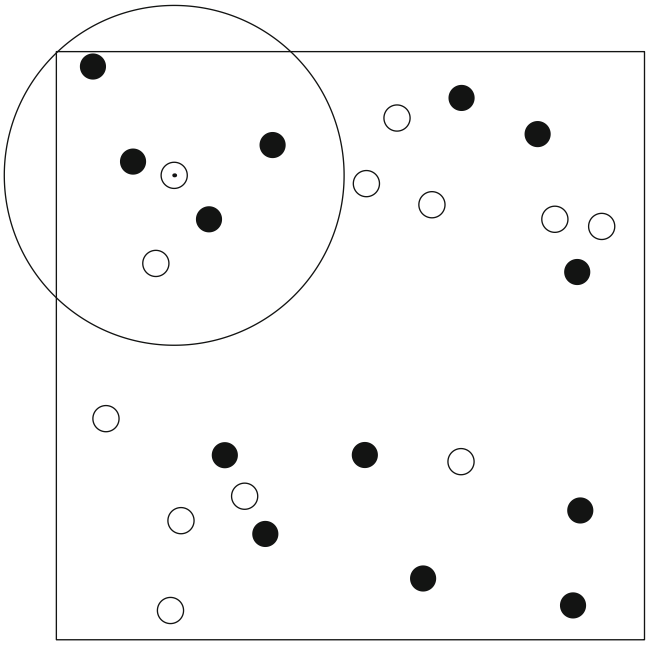


Fig. 5.2 Simple example of local sampling in a robot swarm

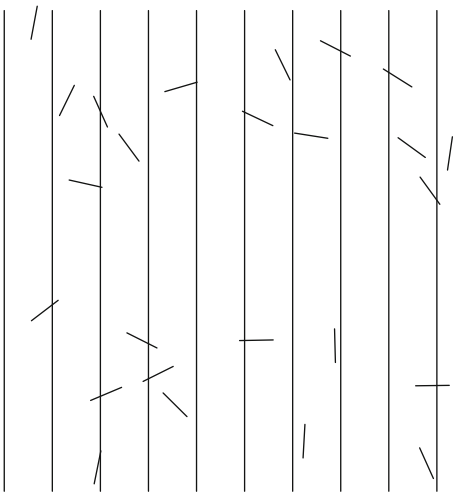


Fig. 5.3 Example configuration of Buffon's needle. Length $b = 7$, line spacing $s = 10$, sample: 14 intersections and 11 non-intersections giving a probability $\hat{P} = 14/25 = 0.56$ while the theory gives $P = (2 \cdot 7)/(10\pi) \approx 0.446$

The second example is more complex. The task is to estimate the size of an area without explicitly measuring it because we assume this is beyond the capabilities of a swarm robot (i.e., no laser scanner, no camera, etc.). How can that be done? An option would be to measure one side length or two, for example, by moving in a straight line with constant speed and taking the time. Then a square or a rectangular shape of the area is assumed and accordingly calculated. A related approach would be to measure the mean free-path-length, that is, the average time between two collision avoidance events.

It turns out that there is a fascinating example in a natural system of how to solve this task. The observation of the ant species *Leptothorax albipennis* and their search for nest sites has revealed this fact. It is possible that these ants follow a procedure that is similar to what is known as Buffon's needle. How this is related to estimating the size of area will be clear a few lines below. Buffon's needle is a statistical experiment. A needle of length b is dropped randomly onto a plane which is inscribed with parallel straight lines that are s units apart (see Fig. 5.3). We assume $b < s$. The needle has a probability $P = 2b/(s\pi)$ of intersecting a line. We do not present a proof here but you can start additional reading from Ramaley's work [322].

We are interested in the sampling error which can be interpreted as the representativeness of the sample. Only here representativeness depends exclusively on the size of the sample set because there are no individual differences between samples (in contrast to the above example of voter surveys). First, we note that the needle experiment is a binomial experiment: the needle either intersects a line or not. Second, we assume that there is an estimated probability $\hat{P}$ that is based on a limited number n of trials (throws of needles). Now we use the binomial proportion confidence interval to give an estimation of the sampling error. The binomial distribution is approximated with a normal distribution. If we choose a 95% confidence level, then we get

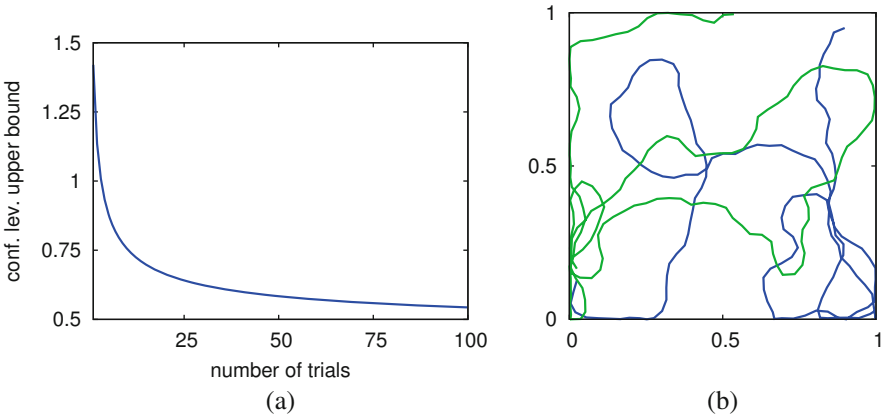


Fig. 5.4 Upper bound of a confidence interval, Eq. (5.5), and two sets of lines. (**a**) Upper bound of a confidence interval following Eq. (5.5) for an assumed correct $\hat{P} = 0.446$ over the number of trials n. (**b**) Two sets of lines and their intersections

$$\hat{P} \pm 1.96\sqrt{\frac{1}{n}\hat{P}(1-\hat{P})}, \tag{5.5}$$

whereas 1.96 is the appropriate percentile of the standard normal distribution (based on a table lookup). The upper bound of a confidence interval following Eq. (5.5) for an assumed correct $\hat{P} = 0.446$ over the number of trials n is given in Fig. 5.4a. It is good news that the interval quickly shrinks with increasing number of trials. Hence, good estimates can probably be achieved with a few samples only.

The behavior of the ants is described by a variant of Buffon's needle which is based on randomly scattered lines, see Fig. 5.4b. Say we have two sets of lines. The first set of lines has a total length of L_1 and the second set of lines a total length of L_2. The number of intersections of lines from different sets is n. The area in which these two sets of lines are placed is then estimated by

$$A = 2L_1L_2/(n\pi). \tag{5.6}$$

The ant behavior is related in the following way. When their old nest is destroyed single ants, called scouts, are sent out to explore a region to find potential nest sites of appropriate size. They prefer flat crevices in rocks. A potential nest site is visited at least twice. On the first visit the scout ant deploys a special pheromone but not on the second visit. The pheromone trail of the first visit defines the line set L_1 and the scout's trajectory on the second visit defines set L_2.

Mallon and Franks [247] report a number of experiments. In one of the experiments an ant colony was given the choice between two nest sites: one of "standard size" and a smaller one of 5/8th (62.5%) of standard size. They chose the standard size nest site 15 times out 15.

Although this method is based on the behavior of a single agent and not on cooperating groups of agents, it still gives a good example of how powerful local sampling can be. Instead of applying high-tech and expensive sensors the Buffon's-needle approach is effective based on simple methods. In addition this approach is also flexible because no special hardware is used to do the measurements. Finally, this approach gives a good example of nature's non-intuitive problem solving in contrast to standard engineering. However, the question remains of how this approach would have been found and designed without knowing the example of the ant behavior.

5.3 Modeling Approaches

In swarm robotics we are still on the search for an appropriate general modeling technique. That is why many different modeling approaches for swarm robotics have been published. None of them is capable of modeling all typical swarm robotics use cases. Hence, when you want to model a particular swarm system then you have to make a selection of the right modeling technique. Which one to use is determined by the essential system features relative to the task of the swarm. Sometimes space can be abstracted but in pattern formation, for example, it is essential. The following list of models is incomplete but hopefully representative.

5.3.1 Rate Equation

A modeling approach, that was popular at least in the beginnings of swarm robotics, are rate equations. The most important early works on rate equations were published by Martinoli [251, 252] and Lerman [235–239]. Rate equations are usually applied to model chemical systems, in particular chemical reactions and their reaction rate. A simple rate equation is $r = kAB$. This equation models the chemical reaction between two reactants. The concentrations of the chemical species are given by A and B. k is the rate coefficient. Say we have a chemical reaction $A + B \rightarrow C$, then an ordinary differential equation (ODE) is defined by the reaction rate

$$\frac{dC}{dt} = kAB, \tag{5.7}$$

for concentrations A, B, and C. Writing Eq. (5.7) down like that is supported by the so-called law of mass action, a fundamental law of chemistry. However, it seems also quite intuitive, for example, once we interpret the concentrations A and B as probabilities of encountering these reactants. Then the product of both corresponds to finding them in one spot, which leads to the reaction that, in turn, happens at rate k.

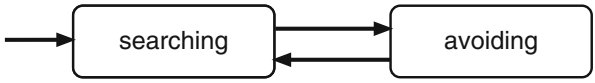


Fig. 5.5 Simple finite state machine for our example on rate equations

We want to model collective robot systems instead of doing chemistry, hence we have to interpret the rate equations differently. Instead of concentrations of chemical species we have swarm fractions of robots that are in certain states. Imagine the robots being programmed by finite state machines and the above A and B would be robots in states A and B of their state machines. Instead of chemical reactions we have state transitions as effects of robot–robot interactions. A state transition in the robot's state machine may be triggered by encountering another robot that is currently in a certain state. This modeling approach is macroscopic, probabilistic, and non-spatial, that is, we live on a well-mixed assumption.

As the basic concept of rate equations is quite simple, we immediately start with a little example taken from Lerman and Galstyan [236]. They want to model a foraging behavior in robots (see Sect. 4.8). In this foraging behavior, robots go for pucks that need to be collected and transported back home. The robots have to avoid obstacles including other robots and initially they wander around and search for these pucks. For an easy start we ignore all the other behaviors necessary for foraging, such as puck detection, homing, entering the home, and reverse homing. We focus on two simple states: searching for pucks and avoiding obstacles/robots (see Fig. 5.5).

To model this simplified system, we define $N_s(t)$ as the number of robots in search state at time t. $N_a(t)$ is the number of robots in avoiding state at time t. We can safely assume a simple conservation law here. The total number of robots stays constant, hence we get

$$N_s(t) + N_a(t) = N. \tag{5.8}$$

Furthermore, we define $M(t)$ as the number of uncollected pucks at time t. We say α_r is the rate coefficient of detecting another robot and α_p is the rate coefficient of detecting a puck. What exactly do these rates define and where do they come from? All specific properties of the robot and how it detects other robots are encoded into α_r and similarly for α_p. Obviously that can be rather tricky but it can be, for example, simple geometric considerations, such as opening angles of sensors and their range. If we change sensors of the robot or we exchange one robot platform for another, then we would like to be able to reflect that in our model. That can be done via these rate coefficients.

We are not done yet as we have to model also some time flow. If a searching robot detects an obstacle, such as another robot or a wall, it executes the avoiding behavior for a time period τ, after which it resumes the search. In a usual implementation of a collision avoidance behavior this time period would actually not be constant, however, for simplicity we assume that here. Now we have to work our way towards actual rate equations. Let's first consider what is expected to happen once two

robots of certain states meet. The number of searching robots N_s decreases when two searching robots detect each other and hence commence avoiding maneuvers. We should also not forget the case when a searching robot detects another robot, that is already in avoiding state, then N_s also decreases. We know that N_s increases when robots return from avoiding state and resume searching. That happens at time t if they started the avoiding behavior at time $t - \tau$, that is, τ time ago. Now we could go through similar considerations for N_a. However, we don't need an equation describing the dynamics of the avoiding robots N_a, because we can compute this quantity using the conservation of the total number of robots N. To write down the rate equations we define fractions of the swarm

$$n_s = N_s/N \tag{5.9}$$

for fraction of robots in state searching and

$$n_a = N_a/N = 1 - N_s/N = 1 - n_s \tag{5.10}$$

for a fraction of robots in state avoiding. These fractions can also be interpreted as concentrations of robots or as probabilities of finding them.

We start with a term for decreasing n_s. It depends on the rate coefficient α_r, if two searching robots approach each other both make a transition, that is, two at a time; and the other possibility was that a searching and an avoiding robot approach each other. Hence, we get

$$-2\alpha_r n_s^2 - \alpha_r n_s n_a. \tag{5.11}$$

Instead of n_a we write $1 - n_s$, expand, sum, and get

$$-\alpha_r n_s^2 - \alpha_r n_s. \tag{5.12}$$

that we write as

$$-\alpha_r n_s(n_s + 1). \tag{5.13}$$

What is missing is the term for increasing n_s. Do we really have to think long about what it should be? Actually not because the amount of robots that return from avoiding state at time t corresponds directly to those that left at time $t - \tau$. The latter case is what we have already modeled, we can reuse the above term (5.13) with inverted signs and the different time $t - \tau$. Once we put everything together including the times, we get

$$\frac{dn_s(t)}{dt} = -\alpha_r n_s(t)(n_s(t) + 1) + \alpha_r n_s(t - \tau)(n_s(t - \tau) + 1). \tag{5.14}$$

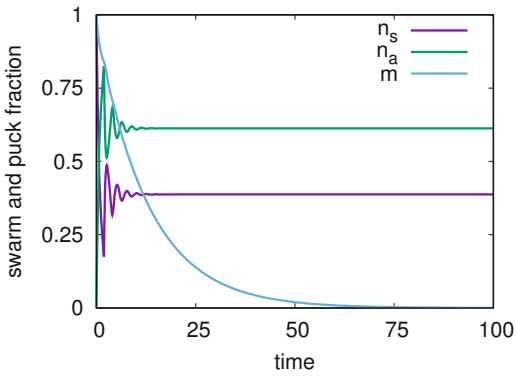


Fig. 5.6 Example for the rate equations approach, numerical solution of an initial value problem ($\alpha_r = 0.6$, $\alpha_p = 0.2$, $\tau = 2.0$, $n_s(0) = 1$, $m(0) = 1$)

This is actually a delay differential equation, that is generally rather difficult to solve. However, that should not bother us here because we can easily use simple numerical solvers. Similarly to n_s and n_a we can also define a fraction of yet uncollected pucks $m(t)$ and define a rate equation for the pucks as well

$$\frac{dm(t)}{dt} = -\alpha_p n_s(t) m(t). \tag{5.15}$$

We can use Eqs. (5.14) and (5.15) to define an arbitrary initial value problem as example. Say we define $\alpha_r = 0.6$, $\alpha_p = 0.2$, and $\tau = 2.0$. As initial values we set $n_s(0) = 1$ (all robots in search) and $m(0) = 1$ (no puck collected yet). We can solve the equations simply numerically [316]. As a result we get a model prediction for the swarm and puck fractions as shown in Fig. 5.6.

The initial oscillations for $t < 15$ may look funny but they should actually be expected for delay differential equations and it is likely that they can be observed in average in swarm robot experiments. Also note the full symmetry between n_s and n_a because of our conservation law $n_s = 1 - n_a$. We end our little study here but for the foraging scenario of Lerman and Galstyan [236] we would need to model many more states.

5.3.2 Differential Equations for a Spatial Approach

The above rate equations approach uses differential equations to model the dynamics of swarm fractions. These are time derivatives. Now, we investigate differential equations to create spatial models, that is, swarm models representing space explicitly. Differential equations allow to represent space continuously, that is, we achieve much higher precision with them than with popular but questionable grid-world models. We use the mathematical tool of differential equations here, in particular stochastic differential equations (SDE) and partial differential equations (PDE).

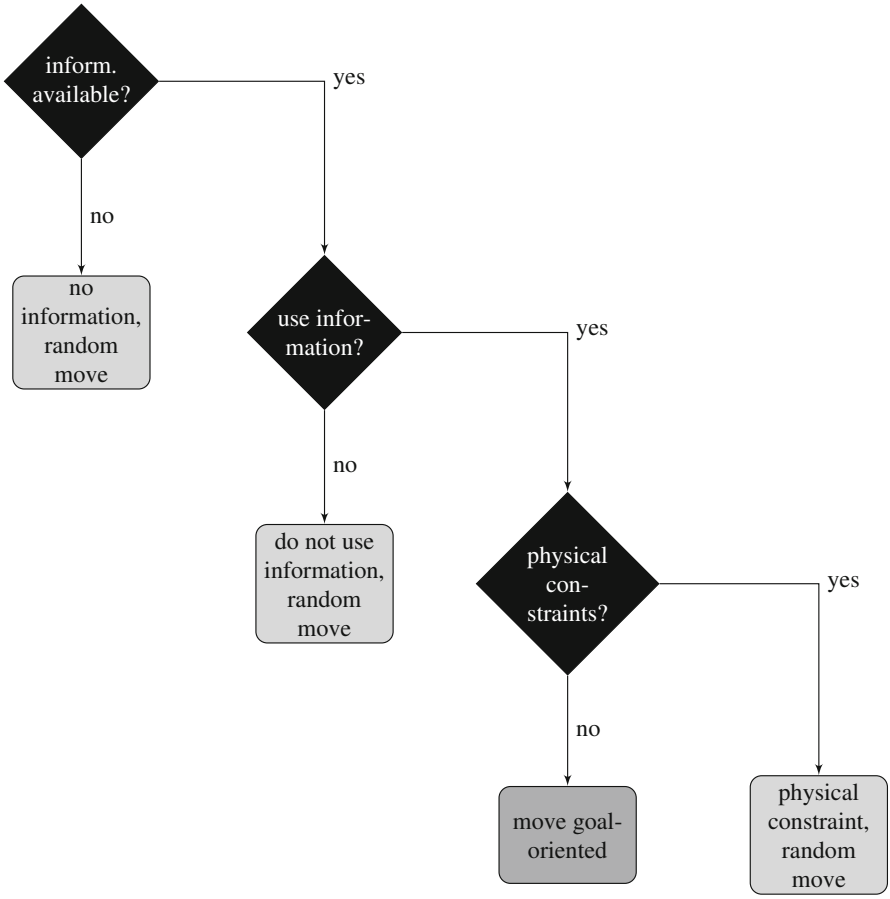


Fig. 5.7 Classes of random robot motion [165, 176]

Before we start, we have first to look into a peculiarity of modeling motion of swarm robots. Following Rosenblueth et al. [330], behavior of biological systems but also of robots can be classified in several ways. We can distinguish active and passive behavior: a robot actively locomoting or a robot floating in a river. The behavior can be purposeful or random: a robot driving into a room in order to explore it or a robot turning away from a wall with a random turn. The purposeful–vs–random distinction can be done even more finegrained [165, 176] as shown in Fig. 5.7. Only a swarm robot provided with proper information has the option of behaving purposeful, otherwise it has to do a random move. If provided with information, it can still choose to ignore the information and do a random move. Finally, even with information and the intention of using it, physical constraints may require to do a random move, for example, to avoid collisions. Hence, modeling purposeful and random behavior of our swarm robots explicitly may turn out to be useful.

We start our journey with a model that for now only represents a single robot. To motivate this approach we interpret a robot's trajectory as a sequence of variables X_t, that is, we assume a discrete time model for the moment. In addition, we say that the swarm robot moves rather erratically, such that it may be fair to say it moves rather randomly. Then we can model the time series X_t as a sequence of random variables. So we have a collection of random variables indexed by a set T representing time:

$$\{X_t : t \in T\}. \tag{5.16}$$

If we assume for now that the robot's motion is completely random, then we can use a fully stochastic model to represent its random motion. Say the robot position at time t is $R(t)$. We model the robot's trajectory with a stochastic differential equation

$$\frac{dR(t)}{dt} = \dot{R}(t) = X_t, \tag{5.17}$$

where X_t is a stochastic process and it is a random displacement of the robot (a little push). However, this is not a good model because usually our robots do not move completely randomly. We improve our approach using a stochastic model with drift. We add a deterministic term, for example, a constant c

$$\dot{R}(t) = X_t + c. \tag{5.18}$$

Our robot is still subject to random displacements in each time step but also constantly makes progress in a given direction defined by c. For example, we can define a 2-d stochastic model with a drift of $c = 0.1$ in x-direction, where the robot is started at (10, 10). In Fig. 5.8, we show two realizations (also called Monte Carlo simulations) of the stochastic process. That is implemented by extending the above equation to the 2-d case with robot coordinates R_x and R_y

$$\dot{R}_x(t) = X_t + c_x, \tag{5.19}$$

$$\dot{R}_y(t) = Y_t + c_y. \tag{5.20}$$

This can be rewritten in vector notation as

$$\dot{\mathbf{R}}(t) = \mathbf{F}_t + \mathbf{C}, \tag{5.21}$$

where we have substituted $\mathbf{F}_t$ for $\mathbf{X}_t$ for later use and apply definitions

$$\mathbf{R} = \begin{pmatrix} r_x \\ r_y \end{pmatrix}, \quad \mathbf{F}_t = \begin{pmatrix} X_t \\ Y_t \end{pmatrix}, \quad \mathbf{C} = \begin{pmatrix} c_x \\ c_y \end{pmatrix}. \tag{5.22}$$

By further extending Eq. (5.21) we can now introduce the Langevin equation. According to the Langevin equation the trajectory of a robot is given by

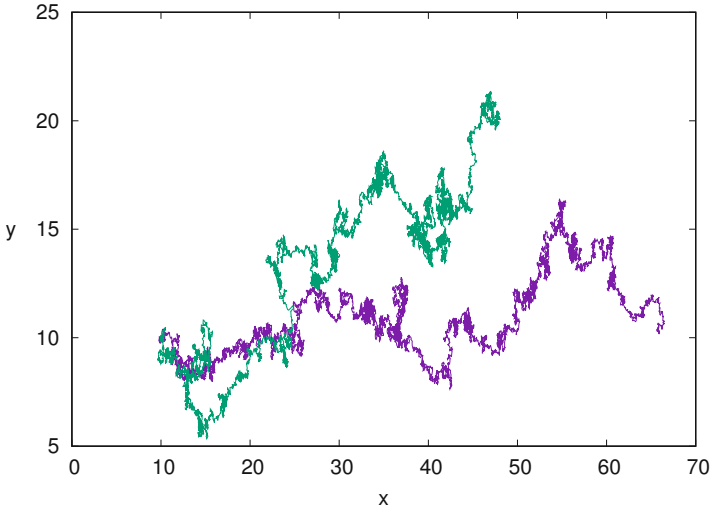


Fig. 5.8 Two realizations of a stochastic model $\dot{R}(t) = X_t + c$ with a drift in x-direction of $c = 0.1$, trajectories are started in (10,10)

$$\dot{\mathbf{R}}(t) = \mathbf{A}(\mathbf{R}(t), t) + B(\mathbf{R}(t), t)\mathbf{F}(t), \tag{5.23}$$

for a stochastic process **F**(t) (e.g., white noise), **A** takes the robot's position $\mathbf{R}(t)$ and time t as input; it describes and scales the robot's deterministic behavior. B describes and scales the non-deterministic behavior. Note that **A** is a vector and hence also gives directional information, while B is scalar and hence only scales the non-deterministic process defined by $\mathbf{F}(t)$. A possible choice of **A** is, for example,

$$\mathbf{A}(\mathbf{R}(t), t) = \nabla P(\mathbf{R}(t), t), \tag{5.24}$$

which defines a gradient ascent in a potential field P. Originally, the Langevin equation was used to model Brownian motion. We use it as a general and generic model to describe the trajectory of a swarm robot.

Next, we have a look at the so-called Fokker–Planck equation. The Fokker–Planck equation is the macroscopically corresponding piece to the microscopic approach described by the Langevin equation. The most fascinating feature of this pair of equations is that they are one of the very few examples of a mathematical connection between micro- and macro-behaviors. The Fokker–Planck equation is mathematically deducible from the Langevin equation, although this deduction is rather involved. The combination of the Fokker–Planck equation with the Langevin equation establishes a direct mathematical micro-macro link.

The equation goes back to the two physicists Fokker [123] and Planck [310]. For physics, the equation is of interest in the connection with Brownian motion, in particular, Brownian motion with drift. Hence, it describes a diffusion process and

if there are no additional assumptions, then we should expect a so-called heat death in the long run for these systems. For example, if we position initially a number of particles close to each other and let them diffuse, then we expect them to approach a homogeneous distribution over time.

Probably the easiest to follow deduction is given by Haken [161] while Risken [329] has dedicated a whole book to the equation. Getting the Fokker–Planck equation out of the Langevin equation requires certain assumptions, such as the stochastic process $\mathbf{F}(t)$ being white noise (i.e., Gaussian distributed and zero mean), a modeled particle needs to have received enough collisions within short time intervals, etc. From the point of view of our application here, this is rather a rare special case or even only an abstract idealistic case. The mathematical difficulty of even this special case gives us a hint of maybe not only the mathematical complexity but also the general complexity of creating micro-macro links.

Now we want to have a closer look. The Fokker–Planck equation is a partial differential equation describing the temporal dynamics of a probability density. This density, in turn, describes in the original physical context the probability of finding the particle within a certain area (e.g., in the 1-d case by integrating it over a considered interval). We are going to interpret it in a slightly different way in the following. The equation is, in direct correspondence to the Langevin equation, a sum of a drift and diffusion term

$$\frac{\partial \rho(\mathbf{r},t)}{\partial t} = -\nabla(\mathbf{A}(\mathbf{r},t)\rho(\mathbf{r},t)) + \frac{1}{2}Q\nabla^2(B^2(\mathbf{r},t)\rho(\mathbf{r},t)). \tag{5.25}$$

ρ is a probability density for a single particle at position $\mathbf{r}$ and time t. In our application of swarm robotics, we interpret it instead as the robot density of all coexisting robots of the swarm. That is, when we integrate over an area W

$$s(t) = \int_{\mathbf{r}\in W} \rho(\mathbf{r},t) \tag{5.26}$$

then we get the expected fraction of the swarm $0 \leq s(t) \leq 1$ within that area at time t. The so-called nabla operator $\nabla = \left(\frac{\partial}{\partial r_1}, \frac{\partial}{\partial r_2}, \ldots\right)$ is based on the spatial derivatives and gives the gradient. It models the "flow" of probability density according to the drift as expected. The drift term pushes density in the direction of the drift $\nabla(\mathbf{A}(\mathbf{r},t)$. ∇^2 is based on the 2nd spatial derivatives and is known as the Laplacian operator

$$\Delta = \nabla \cdot \nabla = \nabla^2 = \frac{\partial^2}{\partial r_1^2} + \frac{\partial^2}{\partial r_2^2} \tag{5.27}$$

and models diffusion. This is also with the expected effect, that is, peaks in the probability density are lowered, the slopes around these peaks get flatter, and the valleys in the density are increased. The constant Q in Eq. (5.25) results from special properties of the stochastic process $\mathbf{F}$, one can say it models the strength of the

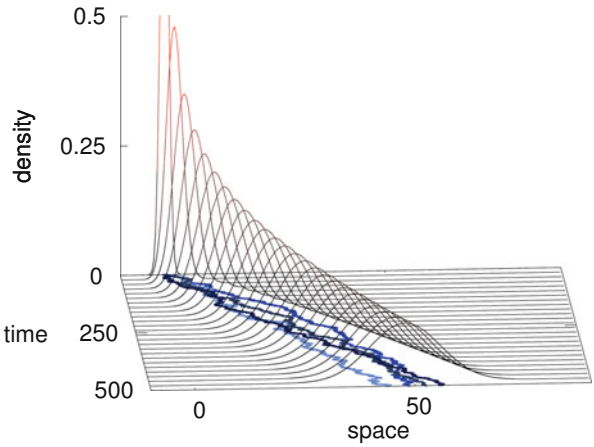


Fig. 5.9 Evolution of a probability density modeled by the Fokker–Planck equation, Eq. (5.25). Example in 1-d space with a drift of $A = 0.1$ and diffusion $B = 0.3$. Shown is the probability density and four example realizations of the corresponding Langevin equation

hits on the particle. The first summand is the drift term and here it models the deterministic part of the robots' behavior. The second summand is the diffusion term and it models the non-deterministic part of the robot behavior.

In Fig. 5.9 we show examples of how realizations of both the Langevin and the Fokker–Planck equation can look like. This figure also clearly indicates the microscopic property of the Langevin equation on the one hand (particular trajectories of robots) and the macroscopic property of the Fokker–Planck equation on the other hand (spatial probability densities of the whole swarm).

The micro-macro duality of the Langevin equation and the Fokker–Planck equation persists also in the application. Say, we would have found appropriate definitions of $\mathbf{A}$ and B, then we could choose whether we want to use a microscopic model (Langevin) or a macroscopic model (Fokker–Planck). For a microscopic approach that would mean to introduce a Langevin equation for each robot to model the robot's motion. The actual simulation is a realization of these multiple, interacting stochastic processes. Robot interactions need to be handled explicitly. For example, a robot could move differently once it has more than five neighbors. With the Langevin approach that can only be done by measuring actual distances between robots and calculating neighborhood sizes.

For a macroscopic approach we need only one Fokker–Planck equation that models the complete swarm at once. Robot interactions can be modeled using the density ρ itself. For example, the above described behavior that is triggered by more than five neighbors can be translated into a minimal density that triggers that behavior. Also note that such a technique inherently introduces a continuum approach. The density increases continuously from values corresponding to 5 robots, to 5.1 robots, 5.2 robots, etc. This continuum should be interpreted as probabilities, such as interpreting 5.1 as "in nine out of ten cases a robot has five neighbors and in one out of ten cases it has six neighbors."

The Fokker–Planck/Langevin agent modeling approach is inspired by "Brownian agents" of Schweitzer [354]. Hamann [165] applied the approach to robot swarms

and Prorok et al. [320] reported a validation by robot experiments. For more details on the Brownian agents, see [114, 186, 353, 355, 356]. Another approach that applies PDE to control robot groups is reported by Milutinovic and Lima [277, 278]. It is an approach strongly influenced by control theory, it makes use of hybrid systems theory, and has a strong theoretical basis.

5.3.3 Network Models

Swarm robotics systems can be interpreted as networks with robots as nodes and edges indicating mutual neighborhood relations. Graph theory provides the more traditional modeling techniques. In addition, there are more recent techniques that can be summarized as "network science." The modeling techniques themselves are similar but focus has shifted to more sophisticated topological features and dynamic networks, where edges are allowed to emerge and vanish over time. In swarm robotics we observe dynamic neighborhoods. A robot may have certain neighbors now but within seconds they can disappear and new neighbors may appear.

5.3.3.1 Random Graphs

A graph G is defined as a tuple $G = (V, E)$, for a set of nodes V and a set of edges E. In swarm robotics, nodes represent robots and edges represent a communication connection or that robots are mutually within sensor range. Being mobile robots, our robots move and hence the nodes should move as well. However, in graph models that is usually not supported. Instead graphs represent rather a snapshot of the system configuration at a given time. Furthermore, we are usually not interested in one particular graph configuration or topology but rather general properties for a given feature, such as the swarm density. A given swarm density would induce a certain probability of having an edge between two robots/nodes. To investigate such general properties, we generate graphs just randomly but with given parameters.

The first model of random graphs was published by Erdős and Rényi [115]. An Erdős–Rényi random graph is an undirected graph $G(X; p)$ with a node set $X = \{0, 1, 2, \ldots, N\}$. Each possible edge is included independently with probability p. With probability p we can emulate the density of the swarm. If p is close to one, many edges are added to the graph and the swarm is dense. If p is close to zero, few edges are added to the graph and the swarm is sparse. The influence of edge-adding probability is non-linear, for example, if we observe the size of largest connected components in the graph. For small p the largest connected component is small. With increasing probability p the size of the largest connected component increases rapidly until the whole graph is part of the largest connected component. For $N \to \infty$, this rapid increase becomes a phase transition, that is, a discrete jump from almost no nodes are part of the largest connected component to almost all nodes are part of the largest component.

A drawback of random graphs is that they do not represent any spatial structure of the swarm. There is no spatial correlation between nodes. For example, if node A is connected to node B and node B is connected to node C, then it seems likely that node A could also be connected to node C.

Geometric graphs fill this gap and explicitly represent space. A geometric graph is defined as an undirected graph $G(X; r)$ and with node set $X \subset \mathbb{R}^d$. In the case of 2-d, we have $X \subset \mathbb{R}^2$ and nodes $x \in X$ and $y \in X$ are connected by an undirected edge if $||x - y|| < r$ for a norm $|| \cdot ||$ on $\mathbb{R}^2$. The norm basically measures the distance between nodes (e.g., Euclidean distance). By the inequality $||x - y|| < r$ we implement the so-called unit disc model, that is, we assume that a robot's sensor range is symmetrical (a disc). For example, in the case of radio communication it is known to be very different from reality [4, 436]. For our purpose here, it is, however, a sufficient model.

Again, we turn to random graphs, only this time random geometric graphs. A random geometric graph is generated by sampling X from a probability distribution, for example, a uniform distribution. As an example, in a 2-d scenario we sample randomly uniformly two coordinates $x_1 \in [0, 1]$ and $x_2 \in [0, 1]$ forming $x = (x_1, x_2)$. Edges are determined, as described above, by checking distances between nodes x and y via $||x - y|| < r$. An example is given in Fig. 5.10. Then the edges are not independent anymore as they are in Erdős–Rényi random graphs. Instead we have spatial correlations in the form "edges $e_1 = (A, B)$ and $e_2 = (B, C)$ make edge $e_3 = (A, C)$ more likely" as discussed above. That makes random geometric graphs a more suitable network model for applications of swarm robotics. With random geometric graphs we can investigate features relevant to swarm robotics, such as average sizes of connected components (groups of robots that cooperate), average number of isolated nodes (robots without any chance to cooperate), and average

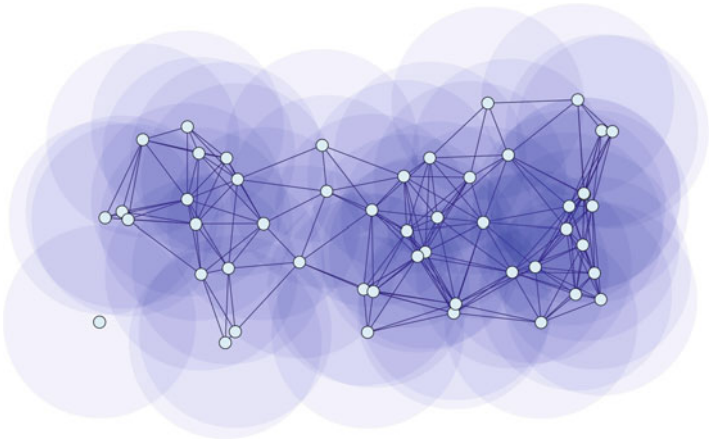


Fig. 5.10 A random geometric graph, blue circles (or disks) indicate sensor range, an edge exists if a neighboring node is located on that disk (created using a script of Friedrich Große)

node degree (possibly a measure for robustness). Random geometric graphs are static models, hence can only represent snapshots of swarm configurations and in particular initial conditions.

5.3.4 Network Science and Adaptive Networks

We summarize models that go beyond random graphs and random geometric graphs as network science here. Network models focus more on specific features of the topology, such as complex networks and small world networks [366, 407]. To stay concise here, we skip most of the general network models that have applications across all fields of research. Instead we focus on a particular sub-class called adaptive networks as defined by Gross and Sayama [156]. Adaptive networks differ from most other network models because they allow dynamic updates of their topology. An example application of adaptive networks is the work by Sood and Redner [360] on collective decision-making using the voter model (see Sect. 6.5.2) on heterogeneous graphs.

Here we closely follow the study of Huepe et al. [191]. They present a simple adaptive network model for the locust experiments [59, 431]. Huepe et al. [191] model network motifs, that are sub-networks, in this case, with up to four nodes. We introduce variable x_L for left-goer density (i.e., fraction of the swarm in state left-goer) and variable x_R for right-goer density. Similarly, we introduce variables for densities of undirected edges (i.e., we do not distinguish $R-L$ from $L-R$) between all combinations of pairs of left- and right-goers: x_{RR}, x_{RL}, and x_{LL}. We do the same for triples: x_{RRR}, x_{RRL}, x_{RLL}, and x_{LLL}. And the same for quadruplets that we notate as $x_{R\cdot L\cdot RR}$, $x_{R\cdot R\cdot LL}$, etc. with the considered node framed with dots $(\cdot)$.

Next we introduce parameters that model the topological changes [191]. $R-L$ links are added randomly at a rate a_{RL} per node. $R-L$ links are deleted randomly at a rate d_{RL} per link. Links between nodes moving in the same direction are added randomly at rate a_{RR} per node and deleted randomly at rate d_{RR} per link. A node switches direction with probability s_{RL} for each $R-L$ link. A node switches direction with probability s_{RRL} for each $L-R-L$ and $R-L-R$ chain. Finally, we introduce noise by a probability s_{noise} that models agents switching directions spontaneously.

The model is written down as a system of ordinary differential equations similar to the rate equations approach, see Huepe et al. [191] for details. For x_R we get

$$\frac{d}{dt}x_R = s_{\text{noise}}(x_L - x_R) + s_{RRL}(x_{RLR} - x_{LRL}). \tag{5.28}$$

For x_{RR} we get

$$\begin{aligned}\frac{d}{dt}x_{RR} = {} & s_{\text{noise}}(x_{RL} - 2x_{RR}) + s_{RL}(x_{RL} + 2x_{RLR} - x_{RRL}) + s_{RRL}(2x_{RLR} \\ & + 3x_{R\cdot L\cdot RR} - x_{R\cdot R\cdot LL}) + a_{RR}x_R^2 - d_{RR}x_{RR},\end{aligned} \tag{5.29}$$

and similarly for x_{LL}. Instead of writing an explicit equation for x_{RL} we write a sum of all two-node motifs

$$\frac{d}{dt}\left(x_{RL}+x_{RR}+x_{LL}\right)=a_{RL}x_Lx_R+d_{RL}x_{RL}+a_{RR}x_R^2+a_{LL}x_L^2-d_{RR}x_{RR}-d_{LL}x_{LL}. \tag{5.30}$$

The above ODE system can be closed and simplified by using the following approximations for triplet and quadruplet motifs (see Huepe et al. [191] for details)

$$x_{RLR}=\frac{x_{RL}^2}{2x_L}, \tag{5.31}$$

$$x_{RRL}=\frac{2x_{RL}x_{RR}}{x_R}, \tag{5.32}$$

$$x_{R\cdot L\cdot RR}=\frac{x_{RL}^3}{6x_L^2}, \tag{5.33}$$

$$x_{R\cdot R\cdot LL}=\frac{x_{RL}^2x_{RR}}{x_R^2}, \tag{5.34}$$

and similarly for the symmetric variables (L and R exchanged). Hence, we end up with a system that models only single nodes and pairs of nodes.

See Fig. 5.11a for the results of a numerical integration of the above equations for initial condition $x_R = 0.55$ and $x_L = 0.45$. Shown is the evolution over time until convergence. See Fig. 5.11b for a bifurcation diagram of x_R over parameter a_{RL} (rate for adding $R-L$ links randomly). The system stays undecided for $a_{RL} < 0.5$. With this simple population-based network model and non-spatial approach, we can analyze the parameters and detect when the system tends to make a collective decision. Huepe et al. [191] have also analyzed for how long the system is expected to stay in a decided state.

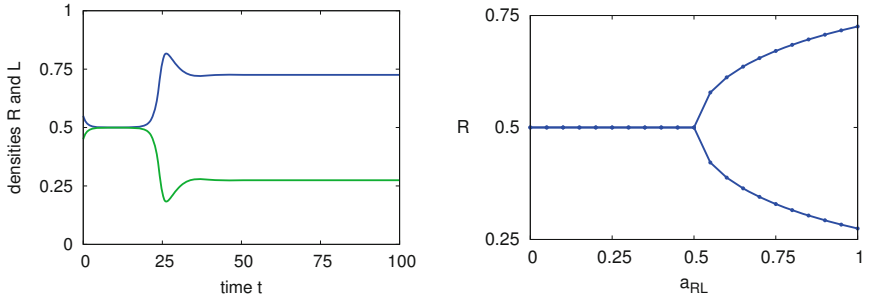


Fig. 5.11 Adaptive networks population model, numerical results for the evolution over time of densities and a bifurcation diagram; parameters: $d_{RL} = 0.25$, $d_{RR} = 0.1$, $s_{RRL} = s_{RL} = 0.2$, and $s_{\text{noise}} = 0.1$. (**a**) Evolution over time of densities x_R and x_L ($a_{RL} = 1$) (**b**) Bifurcation diagram for x_R over parameter a_{RL}

5.3.5 *Swarm Robots as Biological Models*

Instead of only stealing ideas from biology and developing bio-inspired methods in robotics, we should also think about giving something back. There is some research on using robots or even swarm robots as biological models. This rather amazing idea is motivated by using robots as real-world models. Biologists, especially in behavioral biology, formulate hypotheses that need to be tested. Testing is not always possible with animal experiments, in particular when one needs to falsify a behavioral model. Abstract mathematical or computational models as well as simulations may be efficient but have limited credibility and may miss important features of reality. Advantages of robots as "real-world models," hence, include the following. A robot operates in the real world and the roboticist Rodney Brooks once famously said that "the real world is its own best model." Once you have proved your point in the real world it is hard to argue that you missed something (unlike in a simulation). The choice of using robots positively constrains the choice of available models excluding anyway invalid options [409]. It helps to force you to be concrete in specifying the complete biological system and helps you to produce actually testable hypotheses [408]. Using robots allows to study complex agent–environment interactions.

An example study is that of Webb and Scutt [410]. They investigate a behavior of crickets. The male crickets sing their cricket song and female crickets recognize these songs and approach male crickets. This is called "phonotaxis" (i.e., following a sound). "A cricket song typically consists of short bursts [...] of fairly pure tone [...] grouped in various distinctive patterns" [410]. Crickets have one ear in each foreleg and can determine direction-dependent differences between them. The female crickets show a more complex behavior instead of just turning towards the side with the strongest response. Proposed behavioral models are sophisticated and based on known facts from cricket neurophysiology. Webb and Scutt [410] compare a former, complex model that relies on a network that compares the signal amplitude at the two ears. The signal is also analyzed using a low-pass and high-pass filter. This was compared to a simpler model that is effective due to relative latencies in firing onset and "leaky integration" acting as a low-pass filter making fast temporal patterns appear continuous. This was tested on a robot and it may seem reasonable that a physical model of sound propagation, perception, and processing is more reliable than any simulation. The simpler model proved to be effective, hence, questioning the more complex model.

5.4 Formal Design Methods

The design of swarm robotics systems is challenging [165] although the individual behaviors are simple. The designer has to program the globally defined task virtually "between the lines" into the individual robots with their local perception. Designing

adaptive group behaviors is known to be difficult [256] and similarly the design of emergent behaviors [255, 364]. To avoid designing swarm systems with a naive trial-and-error approach, we have to support the designer in any way during the design process. Options are models that predict the to-be-expected behavior for a given control algorithm or, if possible, even an automatic design process.

In general there are two big classes of design approaches for any engineered system; one can design a system either in a top-down or bottom-up approach. Crespi et al. [79] specify:

> In the top-down approach, the design process starts with specifying the global system state and assuming that each component has global knowledge of the system, as in a centralized approach. The solution is then decentralized by replacing global knowledge with communication. In the bottom-up approach, on the other hand, the design starts with specifying requirements and capabilities of individual components, and the global behavior is said to emerge out of interactions among constituent components and between components and the environment.

During the design of a swarm system one typically alternates between these two perspectives. The task is defined globally on the macroscopic level but the robot behavior and its requirements are defined locally on the microscopic level.

In the following we discuss multi-scale modeling approaches that address the micro-macro problem. We shortly go through options of generating swarm behaviors automatically. These methods are, however, diverse and complex and subsequently a detailed discussion is out of this book's scope. Then we discuss applications of classical software engineering in the swarm domain. Finally, we present some of the few formal approaches of programming robot swarms.

5.4.1 Multi-Scale Modeling for Algorithm Design

Even in an unsophisticated trial-and-error approach one would usually not test the control algorithm directly on the robots. Instead we would study the generated swarm behavior in a simulation. Hence, we already have a multiscale approach as we test our software in an abstract model first and then in the real system (i.e., a system of maximal detail). In addition to the scale "level of detail" we also have the scale micro and macro. As mentioned above the designer necessarily has to alternate between the micro and the macro level. The levels of detail are not restricted to two (simulation and reality), but can be a whole cascade of increasing complexity as shown in Fig. 5.12. An immediate sophistication of the mere trial-and-error approach can therefore be an iterative multi-level or multi-scale approach. The initial algorithm design is tested only in a very abstract model until the designer is satisfied. Only then we move to the next more detailed level, adapt the algorithm again until it works properly, and move on. The hope in this approach would be that only small incremental adaptations are required from step to step, hence, simplifying the overall approach. Going instead directly from an abstract model to reality may introduce fundamental issues that may require heavy changes to the algorithm.

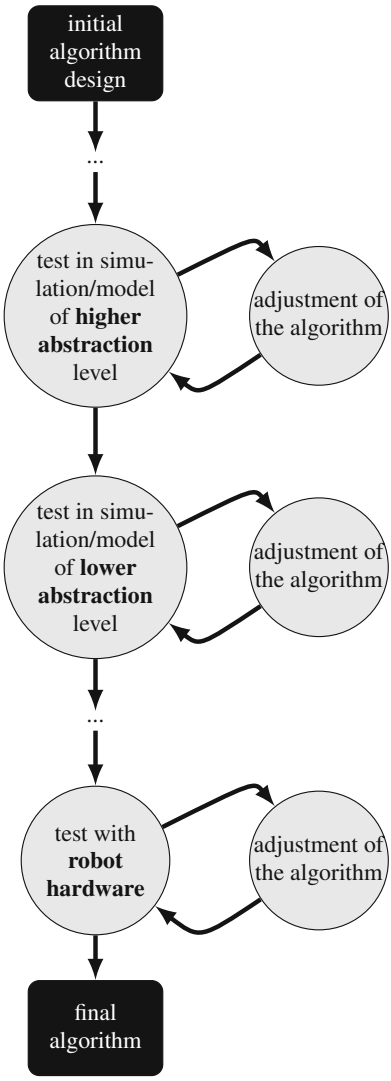


Fig. 5.12 An approach to iterative algorithm design based on a hierarchy of simulations, models, and robotic hardware; schema adapted from [165]

Multi-scale approaches are frequently applied in swarm robotics. Usually they are not as fine-grained as displayed in Fig. 5.12 but rather restricted to only a few or even only two levels. Typically two levels often come as probabilistic microscopic model and deterministic macroscopic model. For example, Vigelius et al. [402] use a Markov process on the micro level and a Fokker–Planck equation combined with a master equation on the macro level (in addition to simulations and robot experiments). Valentini et al. [395] use Markov chains for the micro level and ODE systems on the macroscopic level. Also the combination of a Langevin equation

on the micro level and a Fokker–Planck equation on the macro level follows this probabilistic–deterministic dichotomy [165], as well as the rate equations approach (probabilistic state machines and ODE system).

5.4.2 Automatic Design, Learning, and Artificial Evolution

The automatic design of robot controllers for swarms is out of this book's scope but we quickly go through the most important approaches. Given that designing swarm robot controllers is challenging, it is reasonable to check automatic approaches that may even treat the whole problem as a black box. The toolbox of artificial intelligence and specifically machine learning is full of options. So let us look into learning. There are several learning techniques, one of them is reinforcement learning. Reinforcement learning allows for unsupervised learning, that is, the robot learns by itself and does not require a carefully prepared labeled training data set. Instead one defines only a reward function that assigns utility values to certain system states, hence, telling the robot which states are desirable and which need to be avoided.

While learning for single robots is already relatively challenging [337, 370], multi-robot learning is even more so [257, 414]. The problem is that the robot needs to remember pairs of experienced states and chosen actions, in order to exploit previously tested successful actions and to efficiently explore effects of new actions. While the robot is learning, it is therefore building a data structure of these pairs. This does not necessarily have to happen explicitly (i.e., really storing each state–action pair), but can also be done with approximation techniques [415]. Still, in a multi-robot setting the state–action pairs either combinatorically explode with increasing swarm size (exponentially more combinations of actions with each additional robot) or the other swarm members are treated as environment, which makes the learning task itself more difficult.

The work by Yasuda and Ohkura [430] is an example of applying reinforcement learning in a multi-robot setup with actual robot experiments. They learn robot controllers for a collective transport task (cf. Sect. 4.5).

Other learning techniques include, for example, training artificial neural networks (ANN) by the backpropagation algorithm [334]. This is, however, a supervised learning technique that requires labeled training data. Acquiring labeled training data in robots is almost impossible. Similarly, the currently hyped techniques of deep learning cannot be applied, at least not in an intuitive straightforward way.

Besides standard learning techniques, there are so-called evolutionary algorithms [109]. They are originally optimization techniques roughly inspired by Darwinian evolution and natural selection. Their application to robotics is called evolutionary robotics [50]. In the standard approach, the idea is to use ANN but these are not optimized by learning. Instead one defines a fitness function, similarly to the reward function in reinforcement learning, to evaluate an observed robot

behavior. Initially, one creates a population of randomly generated ANN, that are then tested in a robot simulation or even on the real robot. The fitness function should detect early successes in the population, these ANN are then selected, recombined, and mutated. A new population is created and the algorithm is iterated over many generations. Although it may seem counterintuitive of how random changes to ANN could result in useful robot behaviors within finite time, the approach is successful [293].

The application of evolutionary robotics to swarm robotics is called evolutionary swarm robotics and can be implemented in a rather straightforward way [383]. Instead of uploading one ANN to one robot and testing it in an evaluation, the same ANN is uploaded to all swarm robots and the whole swarm is evaluated. So the standard approach is a homogeneous approach with all robots having the same ANN (but different instances of it). The heterogeneous approach is possible, that is each robot evolves its own ANN, but much more challenging.

There are many successful studies of evolutionary swarm robotics. Examples are the evolution of aggregation behaviors [384] and the evolution of direct robot–robot communication [386], which usually turns out to be very challenging. Instead of ANN, it is also possible to use other representations of robot controllers. For example, finite state machines were used by König et al. [220] in a navigation task. There are also more sophisticated algorithms going beyond the standard methodology of evolutionary algorithms. An example is "novelty search" by Lehman and Stanley [230, 231], which replaces the fitness function with a desire to explore the unknown (i.e., new robot behaviors that had not be seen before are more desirable). Gomes et al. [148] apply this technique to swarm robotics and evolve aggregation behaviors and resource sharing behaviors. Ferrante et al. [118, 119] evolve task specialization and task partitioning with standard evolutionary algorithms but a system of behavioral rules instead of ANN.

Francesca et al. [125] present a dedicated approach to the automatic design of robot swarms based on predefined behavioral modules and probabilistic finite state machines. In their paper, Francesca et al. [125] compare the performance of their automatic design approach to manual design of robot behaviors in five tasks and find advantages for the automatic approach.

5.4.3 Software Engineering and Verification

There are studies of software engineering for agent-based systems called agent-oriented software engineering (e.g., see Lind [241]). They are, however, rather focused on software agents. There are only few software engineering approaches with specific focus on swarm robotics. A rather rough engineering methodology for swarms is given by Parunak and Brueckner [303]. In the same year Winfield et al. [419] push towards "a new discipline of swarm engineering."

Within software engineering there are works focusing on verification. Verification is the process of checking whether a piece of software meets its specification,

that is, whether it is doing what it is supposed to do. Formal verification is the process of proving the correctness of a software (or algorithm). There are only few studies of formal verification in the context of fully distributed robot systems as we have them in swarm robotics.

Winfield et al. [420] propose an approach of formal specification for robot swarms for the example of emergent taxis [43], also see Dixon et al. [95]. They use a linear time temporal logic, that extends regular logic by a time parameter. The defined logic models microscopic movements of robots (however, discretized) and their sensor input. The approach has potential to help in formulating a sophisticated design methodology for swarm robotics.

Gjondrekaj et al. [145] propose a method for formal verification of robot swarms based on a formal language called "KLAIM" [85]. KLAIM is a coordination language that allows to define specifications as accurate models of distributed systems based on a set of primitives. One defines behavioral modules similar to behavior-based robotics (see Sect. 2.4.2). Each module gets input from sensors or other modules and gives outputs to other modules or actuators. The analysis of the specification can be done quantitatively by using the stochastic variant of KLAIM called STOKLAIM [86]. Then one can assign rates to each action that model duration times of actions. Gjondrekaj et al. [145] use formal tools (stochastic modal logics and model checking) to analyze different performance features of the swarm system. They propose a five-step process: (1) formal model with KLAIM as specification, (2) add stochastic aspects, (3) convert specification into code for the robot, (4) test in simulations, and (5) test on real robots.

Brambilla [53] proposes a top-down approach using property-driven design. The idea is to design robot swarms in a systematic way, such that the swarm will behave correctly by design. He applies methods of model checking and analysis tools, such as Bio-PEPA [72] and KLAIM again. Bio-PEPA is a process algebra originally designed to analyze biochemical networks. It includes tools, such as stochastic simulation, analysis based on ordinary differential equations (ODE), and model checking using PRISM. PRISM is a probabilistic model checker [226]. While the property-driven design approach provides a guideline of how to develop the swarm system, the human designer still needs to be creative in order to make the transition from macro to micro and to implement the actual robot behavior.

The idea of design patterns is followed by Reina et al. [324], for example in scenarios of collective decision-making [325]. The underlying idea is that we currently cannot find generic design methods for swarm robotics but we can define solutions for broad classes of problems, that is, design patterns. Reina et al. [325] define a macroscopic model based on rate equations (see Sect. 5.3.1) forming an ODE system and a microscopic model based on probabilistic finite state machines. A micro-macro link (see Sect. 3.3) is established by connecting micro- and macro-parameters. Guided by these models and the known relationship between them, the idea is that a human designer is fully aware of microscopic design choices and to be expected macroscopic effects.

5.4.4 *Formal Global-to-Local Programming*

With the concept of the macroscopic and microscopic level in swarm robot systems in mind, the idea of global-to-local programming is easy to grasp. The user should only be required to input a global description of the desired swarm behavior and some automatic process would then output executable program code for an individual swarm robot. Following the pattern formation example of Yamins and Nagpal [428], we only specify a desired global pattern, which is then processed by a compiler that outputs local rules for individual cells in a grid (see Fig. 5.13 and Sect. 4.3.1). Writing that concept down is so simple compared to how difficult it is to solve this grand challenge of swarm robotics. However, the simply described but hard problems are the best!

Given that the problem is hard, it seems fine to start simple and simplify as much as possible. That is basically what Yamins and Nagpal [428] are doing, also see [426, 427]. They focus on pattern formation in a one-dimensional grid world, which is more accessible than dynamic robot behaviors. The idea is to define a set of local rules that implement a "virtual self-organizing Turing machine" [428]. The set of rules is defined beforehand making heavy use of human creativity. Simple rules can be defined for "single-choice patterns" that are constrained and, hence, simplify the system's processing of the initial configuration at runtime. If there is anyway only one option of how to form the pattern (e.g., the very left grid cell has to be blue), then this one option is just implemented by the local rules. More interesting cases are, of course, situations when symmetries need to be broken and the grid cells have to collectively agree on which option of the pattern they want to implement today. The symmetry-breaking process is typically sensitive to the initial configuration of the grid but still information about a relative reference frame needs to be propagated through the grid. Yamins and Nagpal [428] use local rules to propagate signals as gradient waves through the grid. With a set of generic local rules at hand, most of the work is already done and the compiler only needs to automatically translate the global pattern into appropriate instantiations of those generic rules.

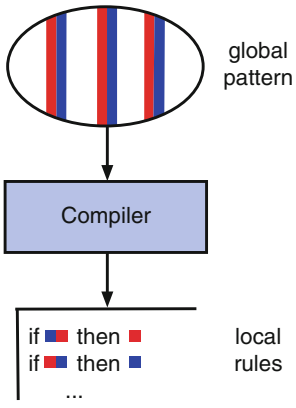


Fig. 5.13 Schema of global-to-local programming following the pattern formation example of Yamins and Nagpal [428]

A similar but more complex approach is followed by Werfel et al. [413] to implement collective construction (see Sect. 4.4). Both the robots and the building material (bricks) are designed such that the whole scenario is simplified to a grid world again, however, three-dimensional now. Again, a user specifies a global pattern; this time it is the desired construction. A compiler needs to translate this blueprint to local rules for the robots. Werfel et al. [413] represent the local rules by labeling grid cells of a 2-d projection with predetermined paths, along which the robots are allowed to travel. This representation is called "structpath." Each grid cell is labeled by at least one arrow pointing to a neighboring cell. This arrow indicates the allowed motion patterns and hence allowed trajectories of the robots. Also several neighbors can be labeled as allowed destinations. A further simplification is that there is one seed block that defines the entrance to the construction side as well as its position and orientation. Based on this structure, the compiler needs to obey a number of constraints. For example, robots cannot climb steps that are higher than one brick and robots cannot put bricks into only one-brick-wide spaces. Deadlocks need to be avoided at all cost. A robot must not get trapped between other robots. For all possible temporal configurations of the construction during the incremental construction process, one needs to avoid that robots block certain areas by placing a brick at a particular spot too early. In summary, the compiler mostly handles these rather basic constraints while the actual micro-macro problem is resolved by the simplifications and restrictions.

5.5 Further Reading

There are not too many options for books on modeling swarm robot systems. One option is the book by Valentini [392], which is however focused on collective decision-making (see Chap. 6). Another option is the book by Hamann [165], which covers the concept of swarm models based on Fokker–Planck equations. Other, more specific options are the book by Galam [133] on sociophysics (opinion dynamics) and the book by Gross and Sayama [156] on adaptive networks.

5.6 Tasks

5.6.1 Task: Beyond Binary Decision-Making

In Eq. (5.4) we gave a modeling abstraction for a binary decision-making system:

$$f(\gamma) = \frac{|\{s_i | s_i = A\}|}{N} = \frac{[\text{number of agents in favor of option } A]}{N} = \varphi. \qquad (5.35)$$

What is the analogon to that for a collective decision-making system with three options (A, B, C)?

5.6.2 Task: Buffon's Needle

Implement Buffon's needle for needle length $b = 0.7$ and line spacing $s = 1$. Note that you do not have to simulate the whole setting in a big 2-d plane. Instead you can reduce the situation to a single line segment of width s because the middle of the needle will always end up between two lines. What remains is simple geometry that determines whether the needle intersects with one of the two lines based on the needle's angle.

(a) Test your program by checking whether it estimates good approximations to π according to $\pi = 2b/(sP)$.
(b) An experiment consists of n needles being thrown. Run experiments with $n \in \{10, 20, \ldots, 990, 1000\}$ trials. For each setting of n, run the experiment 10,000 times (i.e., 10,000×n) and calculate the standard deviation of the resulting set of intersection probabilities. Plot the standard deviation over the number of trials n.
(c) For a maximum of $n = 100$ trials, plot the currently measured probability of many experiments over the trial number n along with the Binomial proportion 95%-confidence interval.[1]
(d) Measure the ratio of experiments for which the true probability[2] is outside of the 95%-confidence interval based on the measured probability. Plot this ratio over the number of trials n. Interpret your results.

5.6.3 Task: Local Sampling in a Swarm

We have a robot swarm of size N with the robots uniformly distributed over the unit square (robot positions (x, y) with $x \in [0, 1]$, $y \in [0, 1]$). Each robot is either black or white with equal probability. The local sampling is done in the following way. A particular robot perceives the color of all robots (including itself) in the neighborhood. The neighborhood is defined by the sensor range r. A robot belongs to the neighborhood of another robot if their distance is less than the sensor range r. From the local ratio of black robots the robot estimates the overall number of black robots in the swarm.

Implement this scenario. Do experiments for swarm sizes $N \in [2, 200]$ each with sensor ranges $r \in [0.02, 0.5]$. For each setting of N and r do 1000 independent experiments and calculate mean and standard deviation of the robot's estimate of black robots in the swarm. Plot mean and standard deviation for some interesting

[1] From Sect. 5.2.2 we know that the Binomial proportion 95%-confidence interval is defined by $\hat{P} \pm 1.96\sqrt{\frac{1}{n}\hat{P}(1-\hat{P})}$.

[2] From Sect. 5.2.2 we know that the true probability is $P = 2b/(s\pi)$.

settings. Try to think of consequences that these measurements would have for an implementation of a robot swarm whose effectivity is directly dependent on these local samplings.

5.6.4 Task: Dimension Reduction and Modeling

We implement the locust simulation (continuous space, discrete time). The locusts live on a ring of circumference $C = 1$ (positions $x_0 = 0$ and $x_1 = 1$ are identical). The locusts move with a speed of 0.001 either to the left ($v = -0.001$) or to the right ($v = +0.001$). They have a perception range of $r = 0.045$. A locust switches its direction in one of two situations:

(1) The majority of locusts within its perception range have opposite direction to that of the considered locust.
(2) A locust spontaneously switches its direction with a probability $P = 0.015$ per time step.

Initially the locusts are uniformly randomly distributed and are moving left or right with equal probability. We simulate a swarm of size $N = 20$ for 500 time steps.

(a) Implement and test your simulation. Plot the number of left-going locusts over time for one run.
(b) We want to take the number of left-going locusts L as our modeling approach. Hence, we implement the dimension reduction of our model by averaging over a number of system configurations and summarizing them in groups of equal left-goer number. Create a histogram of the observed transitions $L_t \to L_{t+1}$ (change in the number of left-goers within one time step) with your simulation by doing 1000 sample runs of 500 time steps each. For example, you can use a 2-d array $A[\cdot][\cdot]$ of integers and an entry of this array $A[L_t][L_{t+1}]$ is increased by one whenever a transition $L_t \to L_{t+1}$ is observed. Plot the histogram.
(c) In addition, count also the occurrences $M[L]$ of each model state L and use these to normalize the histogram entries: $A[i][j]/M[i]$. That way we get approximations of the transition probabilities. Use these approximations $P_{i,j} = A[i][j]/M[i]$ to sample evolutions of L_t over time t. Plot one such trajectory of L. How does this compare to the plot done in (a)?

5.6.5 Task: Rate Equations

We use the rate equations model of searching and avoiding:

$$\frac{dn_s(t)}{dt} = -\alpha_r n_s(t)(n_s(t) + 1) + \alpha_r n_s(t - \tau_a)(n_s(t - \tau_a) + 1)$$

$$\frac{dm(t)}{dt} = -\alpha_p n_s(t) m(t)$$

(a) Use a tool of your choice to calculate the temporal course of this system of ordinary differential equations (a simple forward integration in time can also be implemented from scratch for this system). Notice that we have delay equations. How should the delays be treated especially early in the simulation ($t < \tau_a$)? Use the following setting for the parameters: $\alpha_r = 0.6$, $\alpha_p = 0.2$, $\tau_a = 2$, $n_s(0) = 1$, $m(0) = 1$. Calculate the values of n_s and m for $t \in (0, 50]$ and plot them. Interpret your result.

(b) Now we want to extend the model. In addition to *searching* and *avoiding* we introduce a third state: *homing* (n_h). Robots that have found a puck do a transition to the state *homing* in which they stay for a time $\tau_h = 15$. We assume that for unspecified reasons robots in state *homing* do not interfere with each other or with robots of any other state (assumption: no avoidance behavior for robots in state *homing* necessary). After the time of τ_h they have reached the home base and do a transition back to *searching*. Add an equation for n_h and edit the equation of n_s accordingly. Calculate the values of n_h, n_s, and m for $t \in (0, 160]$ and plot them. In a second calculation, reset the ratio of pucks at time $t = 80$ to $m(80) = 0.5$ and plot the results. Interpret your result.

5.6.6 Task: Adaptive Networks

We numerically solve the ODE system given for adaptive networks in Sect. 5.3.4. Go carefully through Eqs. (5.28)–(5.34), write down the ODE system, and formulate an initial value problem (e.g., $x_R(t_0 = 0) = 0.6$ and $x_L(t_0 = 0) = 0.4$). Implement your own simple numerical integration method (e.g., forward integration in time) or use a standard mathematical software tool to numerically integrate the equations.

(a) Plot the results and recreate Fig. 5.11a. Interpret the results: what can be said about the occurring edges in the network?

(b) Extend your program and do the necessary scans to recreate the bifurcation diagram shown in Fig. 5.11b. You need to integrate your system for different initial conditions to observe all fixed points. Basically you have to run your integration for an appropriate number of time steps and plot the final observed system state. Repeat that for a number of different parameter values. What does the result mean for our collectively deciding swarm robotics system?

Chapter 6
Collective Decision-Making

They exist in loose swarms [. . .]. However, they will unite in moments of danger, or to be more precise, in the event of any sudden change that constitutes a threat to their survival

—Stanisław Lem, The Invincible,

But the amoebas are certainly creative on an individual basis. [. . .] A thought is probably only taken into consideration if the impulse behind it is strong enough, that's to say if enough yrr are trying to introduce it into the collective at the same time.

—Frank Schätzing, The Swarm

the Emperor . . . was interested in having you advance fictionalized predictions that might stabilize his dynasty . . . I ask only that you perfect your psychohistorical technique so that mathematically valid predictions, even if only statistical in nature, can be made.

—Isaac Asimov, Foundations

Abstract We study methods of collective decision-making—an important capability for a swarm to become autonomous.

Collective decision-making is an essential skill for a swarm of robots in order to form an autonomous system also on the macroscopic level. We start with traditional methods to describe decision-making and rational agents. Group decision-making is introduced and we investigate the example of collective motion as a decision-making process. It follows an extensive walk through modeling techniques for collective decision-making, such as urn models, voter model, majority rule, Hegselmann–Krause model, Kuramoto model, Ising model, fiber bundle model, and sociophysics by Serge Galam among other approaches. We conclude with a discussion of hardware implementations of collective decision-making in swarm robotics.

One of the major objectives in swarm robotics is to create autonomous swarms. Autonomy here means to make independent decisions and hence having the

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2_6

possibility of behaving intelligently. The individual robot certainly should be autonomous, however, also the swarm as a whole should be autonomous. Not only an individual can take independent decisions but also the whole swarm. Therefore, collective decision-making is an essential, if not the most essential capability of a swarm.

Each of us knows that decision-making can be difficult. Even if just you for yourself have to decide, say about a certain career choice, it can be agonizing. Decision-making in a small group of people is even more challenging ("Should we go to the lake or to the movies?") and decision-making on the scale of millions of people can be horrifying as we know from our democratic elections. For a swarm of robots decision-making is in some ways easier and in other ways harder. On the one hand, we will program the robots to follow a common goal. Hence, they have a major advantage over human society where we celebrate diversity even though it complicates our decision-making. In terms of a common objective, we synchronize the swarm but, on the other hand, we still face difficulties. In human decision-making, such as elections, we assume that several global channels of communication exist. Of course, not all information is public, not everyone has the same amount of information, and we don't have full transparency but a lot of information is shared globally. In a robot swarm we don't have a global channel, instead robots can only communicate with their neighbors. This creates a number of problems. Robots need to learn that there is a collective decision-making process currently running. Robots may have different information in sub-populations and hence disagree on what the best option is even though they share the same objectives. The swarm may run into deadlock situations, possibly even without noticing it. Finally, the robots may not know whether the decision has been taken already.

In this chapter, we first look into decision-making in general and how it is done in groups. Then we try to understand collective motion as an example for collective decision-making and go through a number of models. Collective decision-making as it is implemented in swarm robotics takes influence from many different fields of research. Usually we explicitly include randomness into the system to ensure a minimum amount of exploration. This makes the swarm system inherently stochastic. Models from statistical physics can represent collective decision-making. Models from the field of opinion dynamics are, of course, relevant. Furthermore, models from material science and models of standardization can also be applied to collective decision-making. While it would be convenient to just pick the best of these options and stick to it, we want to gather a broader knowledge and learn from all of them. The investigation, modeling, and analysis of collective decision-making is still a young field and it will pay off to stay explorative in terms of models at the current stage.

6.1 Decision-Making

Any kind of an autonomous robot has to make a decision at some time. A decision is the selection between possible actions. Decision-making is the cognitive process of selecting an action from a number of alternatives. The perfect decision is an optimal decision which means that no alternative decision results in a better outcome.

Decision-making can be based on certainty, that is, all current circumstances that influence the decision-making are known. The environments in which our robots typically operate are, however, not that simple. Instead we assume that the robot has to decide under uncertainty which is the actual art of decision-making. Decision-making under uncertainty means that the current state and/or the result of the robot's actions are not fully known. In order to still have the chance to act intelligently the robot needs to know preferences between different possible outcomes of its actions. We assume that the robot is able to estimate the outcome of its actions based on the incomplete information about the current state of the environment. Each outcome has a certain utility which is a measure of the quality of how useful a certain outcome is. A rational decision is an optimal decision under a set of constraints such as uncertainty. An agent is rational if and only if it chooses the action that yields the highest expected utility, averaged over all the possible outcomes of that action. This is called the principle of maximum expected utility.

A common way of organizing and representing decision-making are decision trees. An example is shown in Fig. 6.1.

In the following we formulate a general model of decision-making. The robot has m possible alternative actions $A_1, \ldots, A_m$ available and there are n possible initial states $s_1, \ldots, s_n$. The robot knows a utility $N_{ij} \in \mathbb{R}$ for each action A_i and state s_j. These utilities define the utility matrix $N = (N_{ij})_{1 \le i \le m, 1 \le j \le n}$. As discussed above, decision-making under certainty is easy and hence the strategy under certainty about the current state (i.e., j is given) is clear: select $\hat{i} \in \{1, \ldots, m\}$ so that $N_{\hat{i}j} = \max_i N_{ij}$. However, we assume uncertainty that is the robot is not sure about the current state

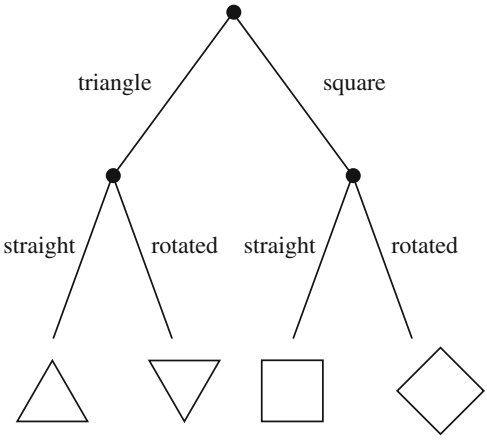


Fig. 6.1 An example of a decision tree (CC0 1.0)

of the environment but might know probabilities of being in a certain state. We define three strategies of decision-making under uncertainty.

Using the cautious strategy (max-min-utility) the robot selects $\hat{i} \in \{1, \ldots, m\}$ so that $N_{\hat{i}j} = \max_i \min_j N_{ij}$. The minimally possible utility (i.e., the assumed worst possible state j) is maximized over all possible strategies i.

Using the high-risk strategy (max-max-utility) the robot selects $\hat{i} \in \{1, \ldots, m\}$ so that $N_{\hat{i}j} = \max_i \max_j N_{ij}$. The maximally possible utility (i.e., assumed best possible state j) is maximized over all possible strategies i.

Using the alternatively cautious strategy (min-max-utility) the robot selects $\hat{i} \in \{1, \ldots, m\}$ so that $R_{\hat{i}j_i} = \min_i \max_j R_{ij}$ whereas $N_j = \max_i N_{ij}$ (maximal utility in state j) and risk matrix $R_{ij} = N_j - N_{ij}$ (risk of action i in state j). The maximally possible risk (i.e., assumed most risky state j) is minimized over all possible strategies i.

6.2 Group Decision-Making

We have learned that decision-making under uncertainty is already a difficult task. In swarm robotics we have, however, multiple robots that cooperate and consequently also have to make decisions collectively. Group decision-making or collaborative decision-making is needed when a group of individuals makes a choice from alternatives. In contrast to a single agent making a decision, group decision-making is more complex and might show effects of social influence and, for example, group polarization.

In consensus decision-making the idea is to avoid "winners" and "losers" of the decision. While the majority approves the decision, the minority agrees to go along with that decision. This is implemented by giving the minority the right to veto against any decision. Obviously this procedure is complicated as everyone can tell who has taken part in a consensus decision-making process within a diverse group.

More common are, of course, voting-based methods. A strict method based on voting is to require a majority (>50%) for a decision, even if there are more than two options. Plurality, in contrast, allows decisions based on less than 50%, that is, just the largest block decides.

There are a number of challenges in group decision-making. An example is "Groupthink" (1972, Irving Janis). Groupthink is a psychological phenomenon within a group of people who minimize conflict and reach a consensus decision without critical evaluation of alternative ideas.

The difference between individual decision-making and group decision-making is that not just one agent decides but that a set of individual decisions exists. How these individual decisions (micro-level) can be generally transformed into a decision of the group turns out to be a general problem.

Each voter is represented by a relation R_i of its own. $N(x, y)$ is the number of voters with xRy. We define the majority relation M by $xMy \Leftrightarrow N(x, y) \geq N(y, x)$. That way M is always reflexive and connex (counting votes always works) but not always transitive.

There are several so-called voting paradoxes. A minimal example of such a voting paradox is the following. We have three candidates $X = \{a, b, c\}$ and three voters $I = \{1, 2, 3\}$. Each voter i specifies her or his preferences by a relation R_i: $R_1 = \{(a,b), (b,c), (a,c), (a,a), (b,b), (c,c)\}$, $R_2 = \{(c,a), (a,b), (c,b), (a,a), (b,b), (c,c)\}$, and $R_3 = \{(b,c), (c,a), (b,a), (a,a), (b,b), (c,c)\}$. These preferences determine a majority relation that has a cycle: $M = \{(a,b), (b,c), (c,a), (a,a), (b,b), (c,c)\}$. The majority is said to prefer a over b, b over c, and c over a, which is obviously a contradiction.

The scientific field that analyzes the above stated problem of how to aggregate individual preferences to achieve a group decision is called social choice theory (also known as theory of collective choice). The list of results from this research includes the above-mentioned voting paradox. Additional results are Arrow's impossibility theorem and the Ostrogorski paradox. Arrow's impossibility theorem states that with more than two candidates no rank order voting system can convert the ranked preferences of individuals into a community-wide ranking while also meeting a specific set of (democratic) criteria. The Ostrogorski paradox refers to the problem of combining preferences in sets. There is a distortion of voters' attitudes when they are voting for collections of factual issues as it is common in the form of party platforms.

6.3 Group Decision-Making in Animals

From the knowledge that group decisions are complex and often paradoxical we infer that decision-making in groups of animals is a challenge. In addition, these animal groups often rely on consensus decisions because an opposing minority might provoke the separation of the swarm which jeopardizes the survival of the swarm. For example, animal groups have to decide which new nest site or resting place to choose, or in which direction they want to move in a flock. Hence, the question arises how animals can possibly achieve consensus decisions efficiently. There are a few simplifying factors. Typically the members of the swarm have no conflicting interests. The well-being of the swarm is of interest to its members. There is also only a soft requirement for optimality. Approximations are often sufficient. The time-requirements are equally soft and hence a slow iterative process might be acceptable.

An example is nest choice in honeybees. Only about 5% of the swarm disperses as so-called scout bees which explore potential nest sites in the vicinity. Once they have found such a potential nest site they return and advertise their findings by dancing to other scouts. The length of these dances is correlated with the quality of a site which generates a positive feedback (good sites are advertised by long dances which are observed by more bees). Eventually the scouts reach a consensus and lead the remaining 95% of the swarm to the respective nest site. These 95% do not contribute to the decision process itself.

A second example is the determination of travel routes in groups of navigating birds. Although the scientific knowledge is limited, it seems that in rather small groups more experienced birds contribute more during the decision-making process. However, also majority decisions might occur. In relatively large groups self-organizing processes seem likely.

Although many different species rely on consensus decision-making processes which have been sufficient to guarantee their survival, these systems are not completely robust and might result in fatal disasters. An example for a bad collective decision is circular milling in ants. In certain species it might happen that the pheromone trails form a loop which traps a group of ants. Unfortunately they are not able to detect this situation and keep circling until eventually they die.

Collective decision-making in natural swarms is based on typical swarm principles such as local communication and agents interact only with direct neighbors or via the environment. Consequently no direct voting process can be implemented that would be similar to voting in a group of human beings. Instead the process is asynchronous because several spatially separated agents have to start to find a decision at the same time. These local decisions based on local perception might cause deadlocks or oscillations in the process. An example of a deadlock is shown in Fig. 6.2.

Because agents have to operate without global knowledge it is possible that in one region of the swarm a subgroup of agents starts to converge on an option that is different from the chosen option of another subgroup at a different region. Based on a statically defined process this might result in a deadlock. This process is similar to crystallization where impurities (particularly line defects) emerge because crystallization starts at several close-by regions. The adaptive behavior of swarms is however different from the static process of crystallization. It is a dynamic and stochastic system and therefore these impurities do not stay for ever. Even a perfect tie with 50% for option A and 50% for option B will eventually be overcome. This is due to fluctuations that disturb the system and drive it out of local optima (undesired states of limited use).

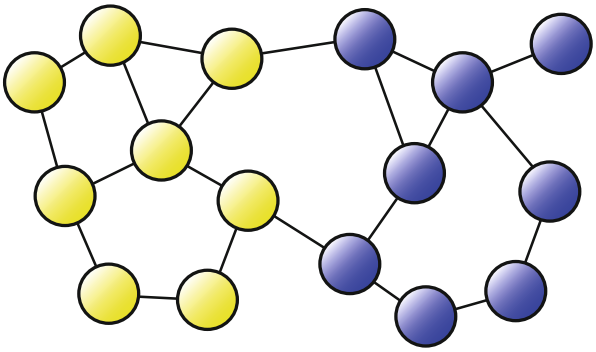


Fig. 6.2 Deadlock situation in a collective decision process of a swarm. Each agent has the correct state following a local majority rule but the resulting global state is an undecided fifty-fifty state

6.4 Collective Motion as Decision Process

Also some collective phenomena that intrinsically rely on continuous features can be reduced to a collective decision. The consensus on a common direction α in a flock can be viewed as a decision out of an infinity of alternatives $\alpha \in [0°, 360°)$. The swarm's motion can also artificially be restricted to a pseudo-1-d setting on a ring (see Fig. 6.3) which leaves only two discrete options: clockwise motion and counterclockwise motion.

As an example we take the behavior of the desert locust, *Schistocerca gregaria*, which shows collective motion, often called "marching bands," in the growth stage of a wingless nymph. The collective motion is density-dependent and individuals seem to change their direction as a response to neighbors [59]. An additional interesting feature is a spontaneous switching behavior that is observed in these swarms. In comparatively small swarms the locusts tend to switch the direction of their collective motion even if they were mostly aligned before.

In the experiments it is observed that groups of locust nymphs (immature form of some invertebrates, particularly insects, which undergoes gradual metamorphosis (hemimetabolism) before reaching its adult stage) with relatively low densities are highly aligned and march in one direction around the ring-shaped arena for up to 2 or 3 h. Then they spontaneously switch their preferred direction within only a few minutes and march in the opposite direction, again for a number of hours. In experiments with higher densities, marching groups travel in the same direction for the full 8-h duration of the experiment.

The essentials of this behavior are grasped by measuring only the percentage of locusts that are in one of the two states (clockwise or counterclockwise motion). These swarm fractions can be measured over a certain period which results in a diagram as shown in Fig. 6.4.

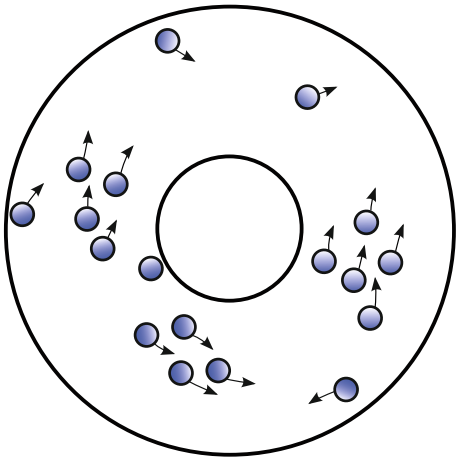


Fig. 6.3 Schematic representation of locusts moving in a ring-shaped arena

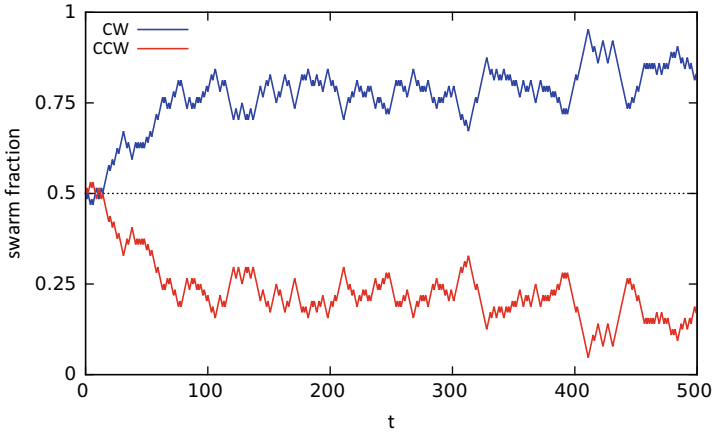


Fig. 6.4 Percentage of clockwise (CW) and counterclockwise (CCW) moving agents over time (discrete time steps of the model)

Yates et al. [431] give the distribution of average velocities over many independent experiments, which is a symmetric bimodal distribution (two peaks, one for a positive and one for a negative speed). It is important to understand that this does not correspond to a symmetric distribution of positive and negative velocities (i.e., clockwise and counterclockwise motion) within a single experiment but gives the average over a number of experiments. Hence, it cannot show the broken symmetry of the direction of motion seen in a particular experiment but instead shows a bimodal distribution.

6.5 Models for Collective Decision-Making Processes

As mentioned in the introduction to this chapter, for modeling collective decision-making in swarm robotics we can take inspiration from many different fields. The one, perfectly generic model seems not yet to exist. Hence, we try to gather a rather broad knowledge about model candidates to enable us to select the most appropriate model for a given scenario. These models differ a lot in their complexity, which means in the ideal case, that we can use very simple models when appropriate.

In the following we will try to reuse a common notation as much as possible. We assume that the swarm has to decide over a set of options $O = \{O_1, O_2, \ldots, O_m\}$ with $m > 1$ options. A robot i has a defined opinion o_i at any time and a neighborhood $\mathcal{N}_i$ of robots. The set $\mathcal{N}_i$ contains all indices of robot's i neighbors and does not include the considered robot's index i itself. A robot i is member of a neighborhood group $\mathcal{G}_i$ of robots. The set $\mathcal{G}_i$ contains all indices of robot's i neighbors and also includes the considered robot's index i itself. In a typical case the task for the swarm is to find a consensus on one of the options O (i.e., 100% agreement in the swarm of one option).

Fig. 6.5 Collective decision-making as process of iterations over three phases: exploration, dissemination, and opinion switch

A collective decision-making process consists of an iteration over up to three phases: exploration, dissemination, and opinion switch (see Fig. 6.5). The options O may have different qualities $q(O_i)$. In a special case all may have the same quality $\forall i, j : q(O_i) = q(O_j)$, hence, the task is a symmetry-breaking problem. Then all options are equal in their utility but the swarm has still to find a consensus on any of them. During the exploration phase a robot i may explore an area associated to its current opinion o_i (e.g., a potential construction site) and gather information about its quality $q(o_i)$.

During the dissemination phase the robot signals its opinion to its neighbors, either by explicit messaging (e.g., via radio or infrared communication) or by giving cues (e.g., lighting an RGB LED). The duration for how long the robot disseminates may vary according to the quality of its opinion. For example, it could be directly proportional to the quality (i.e., doubled dissemination time for doubled quality). This correlation of opinion quality and dissemination duration can then trigger a positive feedback loop. More robots perceive opinions of high quality, they switch to opinions of high quality, and then disseminate high-quality opinions themselves.

During the opinion switch phase, the robots follow a decision-making rule, such as the voter model or the majority rule, to switch their opinion. The robots do not have to follow these three phases of exploration, dissemination, and opinion switch in a synchronized way. Instead each robot can follow its own clock except for keeping the correct duration of the dissemination phase in the case that it is correlated with opinion quality.

Many of the following models are microscopic models, that is, they define a behavior for individual robots. These models cannot be used directly to make predictions about the expected behavior of the swarm. A lot of research actually revolves around that very question of how to determine the global behavior based on a given local decision-making rule. More details about achieving consensus in robot swarms are given by Valentini [392].

6.5.1 Urn Models

In order to get a better understanding of collective decision-making processes like the locust system we develop a model of it. Our guiding principle will be simplicity. Therefore, we focus on a simple binary decision process between options A and B. We say there is no bias between these two options, that is, A and B are of equal utility. Consequently it does not matter on which of the two our system converges as long as it does converge on one. Due to our simplicity principle we restrict the model to a single system variable $s(t)$ which gives, without loss of generality, the swarm fraction that is in favor of option A. Say we start with a perfect tie: $s(0) = 0.5$. What should be our next step? Well, we are interested in how $s(t)$ changes over time. Furthermore, it is our educated guess that this change will depend on s itself which can be described by a function $\Delta s(s(t))$. This function Δs could, for example, be measured in a simulation. However, our objective is to get a better understanding of the underlying principles in collective decision-making systems and hence we want to model a process that defines Δs by itself.

At first, we consider the influence of space in decision-making systems like the locust system. It seems likely that spatial features influence the system behavior. For example, it might be the case that neighboring agents are more likely to share the same opinion than randomly chosen agents. However, we have decided to go for a model with just one variable and consequently we have to develop a non-spatial model. Our model ignores agent positions which means that our model is based on the well-mixed assumption. While one might think that a non-spatial model makes as much assumptions about space as an atheist about God, it turns out that a non-spatial model assumes space to be irrelevant which is equivalent to the assumption of a well-mixed state. There are no correlations between agents based on spatial features. As discussed above this assumption is most likely wrong but we stick to our simplicity motto and willingly accept resulting shortcomings of the model.

6.5.1.1 Ehrenfest Urn Model

Next, we have an intriguing idea. Given we have a well-mixed state and we are working on fractions of a swarm, what about modeling the system as a lottery drawing? In the lottery community well-mixed devices are eminently respected. Hence, we decide to represent robots by marbles in an urn. The distribution of these marbles is following the robot states which is $s(t)$. Urn models are well-known in statistics and they have also been used as models for a variety of systems. For example, there are the Pólya urn models [246, 312] and there is also the famous urn model by Ehrenfest and Ehrenfest [108], which is interestingly also known as the dog-flea model. They followed a serious inquiry as they defined the model to give not less than an explanation of the second law of thermodynamics. Their attempt was to support controversial ideas of Ludwig Boltzmann who sadly committed suicide in the year before. Here is a definition of the drawing process according

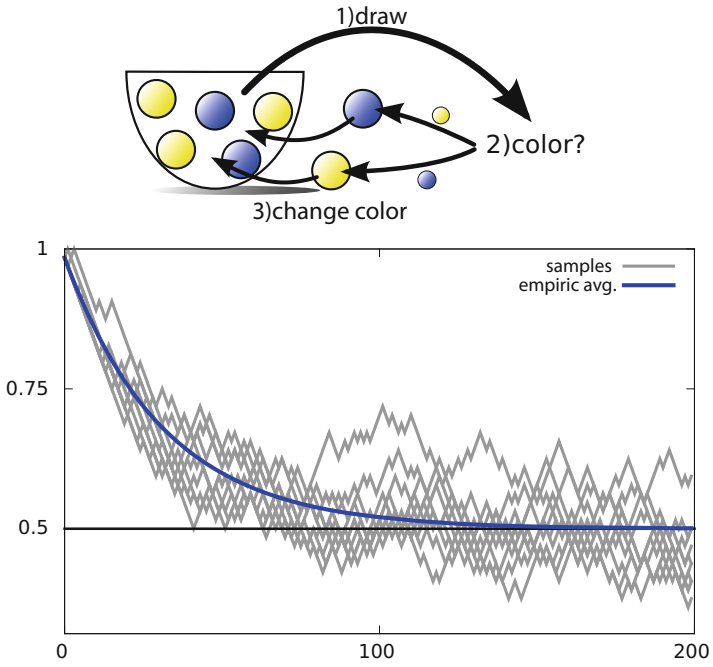


Fig. 6.6 Ehrenfest urn model, top: rules of drawing, bottom: ratio of blue marbles in the urn (vertically) over rounds of iterated drawing (horizontally), initially all marbles are blue

to the Ehrenfest urn model. We have an urn filled with N marbles of two colors: blue and yellow. Say initially all are blue. Now we start drawing. Whenever we draw a blue marble we replace it with a yellow marble. If we draw a yellow marble, we replace it with a blue one (see top of Fig. 6.6).

As an example, we take $N = 64$ marbles and initially all are blue, that is, $s(0) = 1$. Then we do many rounds of the above drawing process. If we keep track of the fraction of blue marbles in the urn over all rounds, we can plot a diagram as shown in Fig. 6.6. The diagram shows a number of independent runs along with an empirically obtained average. The empirical average indicates an exponential decrease. In the following we analyze the system in more detail.

To formalize this process, we keep track of how the number of blue marbles (without loss of generality) changes depending on how many blue marbles were in the urn at that time. We can do this empirically or we can actually compute the average expected "gain" in terms of blue marbles. For example, say at time t we have $B(t) = 16$ blue marbles in the urn and a total of $N = 64$ marbles. The probability of drawing a blue marble is therefore $P_B = \frac{16}{64} = 0.25$. The case of drawing a blue marble has to be weighted by -1 because this is the change in terms of blue marbles in that case. The probability of drawing a red marble is $P_R = \frac{48}{64} = 0.75$ which is weighted by $+1$.

Hence, the expected average change ΔB of blue marbles per round depending on the current number of blue marbles $B = 16$ is $\Delta B(B = 16) = 0.25(-1) + 0.75(+1) = 0.5$. This can be done for all possible states yielding

$$\Delta B(B) = -2 \cdot \frac{B}{64} + 1. \tag{6.1}$$

Hence, the average dynamics of this game is given by $B(t+1) = B(t) + \Delta B(B(t))$.

The recurrence $B_t = B_{t-1} - 2\frac{B_{t-1}}{64} + 1$ can be solved by generating functions [150]. For $B_0 = 0$ we obtain the generating function

$$G(z) = \sum_t \left(\sum_{k \le t} \left(\frac{62}{64} \right)^k \right) z^t. \tag{6.2}$$

The tth coefficient $[z^t]$ of this power series is the closed form for B_t. We get

$$[z^t] = B_t = \sum_{k \le t} \left(\frac{62}{64} \right)^k = \frac{1 - \left(\frac{62}{64}\right)^t}{1 - \frac{62}{64}}. \tag{6.3}$$

Hence, for initializations $B_0 = 0$ and the symmetrical case $B_0 = 64$ the system converges in average rather fast to the equilibrium $B = 32$. The actual dynamics of this game is, of course, a stochastic process which can, for example, be modeled by $B(t+1) = B(t) + \Delta B(B(t)) + \xi(t)$, for a noise term ξ.

As mentioned above the Ehrenfest urn model was designed as a toy model for the application in statistical physics, in particular, to analyze the entropy production of diffusion processes. To complete this short discussion of the Ehrenfest model we check how the entropy of this system evolves over a number of rounds. At first we need a definition of entropy for this system. There are a number of options how to define entropy. Here, we choose the Shannon entropy which we define based on $s(t)$

$$H(t) = -s(t) \log_2(s(t)) - (1 - s(t)) \log_2(1 - s(t)). \tag{6.4}$$

By substituting $s(t) = \frac{1-\left(\frac{62}{64}\right)^t}{1-\frac{62}{64}}$ we get the monotonically increasing entropy as shown in Fig. 6.7. This monotonic increase was proven by Kac [204].

6.5.1.2 Eigen Urn Model

The Ehrenfest model does not serve as a good model for collective decision-making systems because the model converges in average to the undecided state. This is just consequential as it is a model of diffusion while we are interested in self-organizing systems. Fluctuations away from the equilibrium should be enforced. Hence, we basically need to invert the model. It turns out that this was done by Eigen and

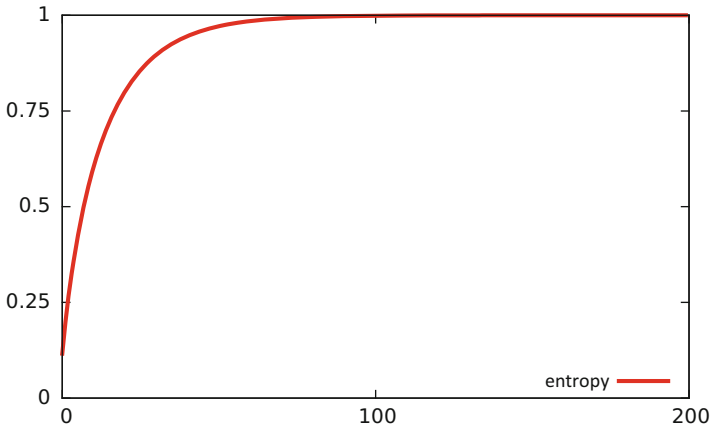


Fig. 6.7 Monotonic increase in Shannon entropy (vertical axis, Eq. (6.4)) over rounds (horizontal axis) for the Ehrenfest urn model

Winkler [112]. Eigen's model was defined to show the effect of positive feedback. It works like this follows. We have an urn filled with N marbles of two colors: blue and yellow. Say initially there are 50% blue and 50% are yellow. In contrast to the Ehrenfest model we draw with replacement now. We draw a marble, notice its color, and put it back into the urn. Whenever we draw a blue marble we replace a yellow one from the urn with a blue marble. If we draw a yellow marble, we replace a blue one from the urn with a yellow marble.

The expected average change of blue marbles per round changes according to

$$\Delta B(B) = \begin{cases} 2\frac{B}{64} - 1, & \text{for } B \in [1, 63] \\ 0, & \text{else} \end{cases}. \tag{6.5}$$

In contrast to the Ehrenfest model, fluctuations are enforced and the distribution of marbles is driven to extremes as shown in Fig. 6.8. The Eigen model has two special configurations. For $B = 0$ and $B = 64$ we can only draw blue ($B = 0$) or yellow ($B = 64$) marbles while we would actually be asked to replace one of the other color from the urn which is impossible in these cases. Therefore, once one of these two states, $B = 0$ or $B = 64$, is reached we stay there forever. Compared to typical collective decision-making systems that seems not appropriate because these extreme situations of perfect consensus are typically not achieved.

6.5.1.3 Swarm Urn Model

To fix the disadvantage of the Eigen model, we develop a variant of an urn model that avoids perfect consensus and is more appropriate for collective decision-making in robot swarms [166]. We start from the Eigen model but include an explicit choice

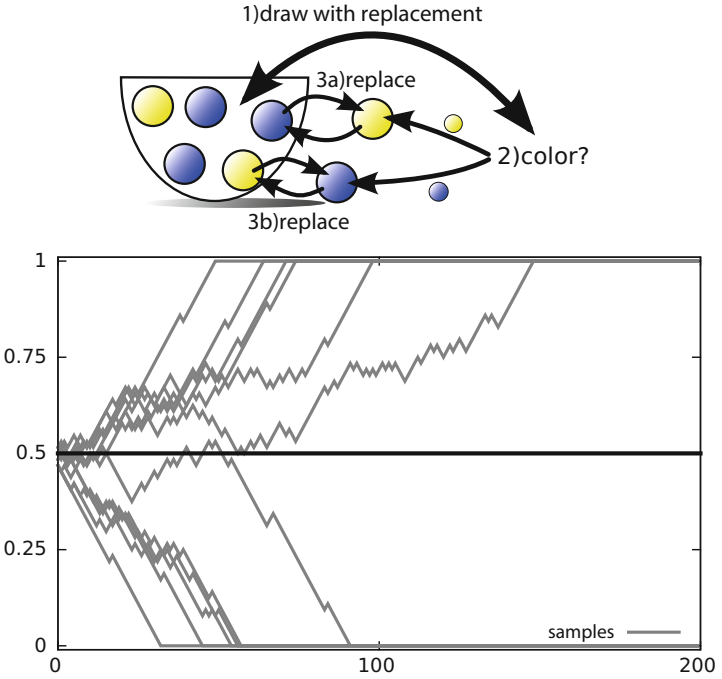


Fig. 6.8 Eigen urn model, top: rules of drawing, bottom: ratio of blue marbles in the urn (vertically) over rounds of iterated drawing (horizontally), initially 50% of the marbles are blue

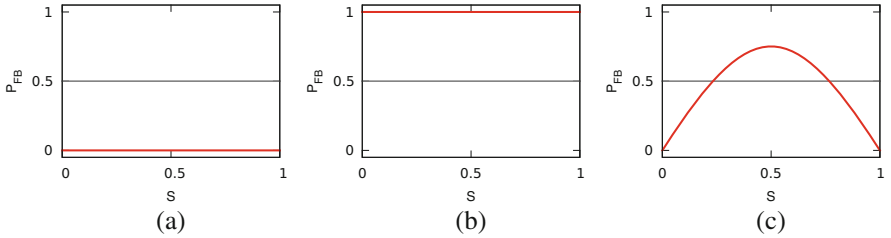


Fig. 6.9 Functions of positive feedback probability $P_{\mathrm{FB}}(s)$ for the (**a**) Ehrenfest ($P_{\mathrm{FB}}(s) = 0$), (**b**) Eigen ($P_{\mathrm{FB}}(s) = 1$), and (**c**) swarm urn model ($P_{\mathrm{FB}}(s) = 0.75\sin(\pi s)$)

between positive and negative feedback. This choice is based on a probability for positive feedback P_{FB}. The drawing round has several steps: draw with replacement, noticing the color, determine whether in this round we have positive feedback (probability P_{FB}) or negative feedback (probability $1 - P_{\mathrm{FB}}$), and a final step that exchanges marbles of the respective colors (see Fig. 6.10, top).

Before we have a look at typical trajectories, we introduce the underlying concept of a positive feedback probability P_{FB}. Considering the Ehrenfest model again, we notice that we have a constant positive feedback probability of $P_{\mathrm{FB}} = 0$ (see Fig. 6.9a). Considering the Eigen model again, we notice that we have a constant

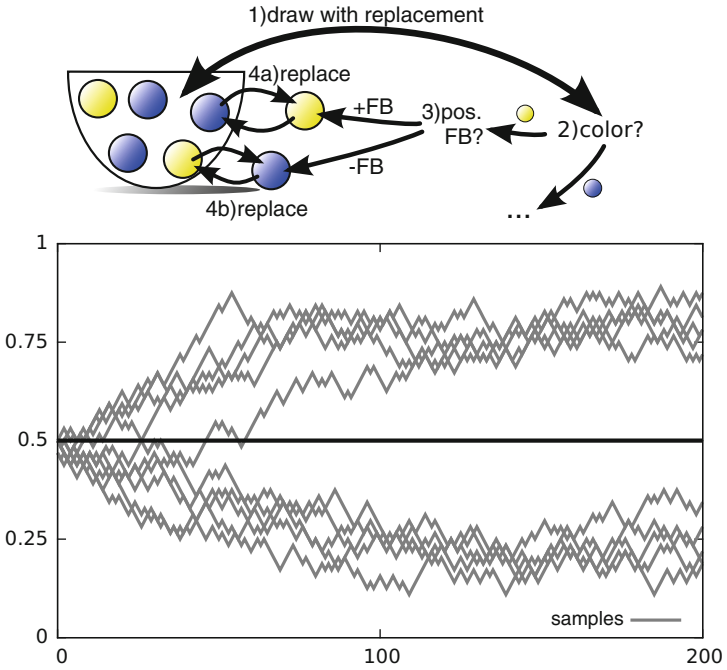


Fig. 6.10 Swarm urn model, top: rules of drawing, bottom: ratio of blue marbles in the urn (vertically) over rounds of iterated drawing (horizontally), initially 50% of marbles are blue

positive feedback probability of $P_{\mathrm{FB}} = 1$ (see Fig. 6.9b). To fix the problem of converging on a consensus in the Eigen model, we need a positive feedback probability that varies depending on the current system state. We need a high rate of positive feedback close to $s \approx 0.5$ to get away from the undecided state based on fluctuations. We need a lower rate of positive feedback, that is, primarily negative feedback close to $s \approx 0$ and $s \approx 1$ to stay away from the consensus states. Out of many possible choices of functions we arbitrarily choose $P_{\mathrm{FB}}(s) = 0.75 \sin(\pi s)$ (see Fig. 6.9c). Examples of resulting trajectories are given in Fig. 6.10 (bottom). As desired the trajectories stay away from the extreme states $s \in \{0, 1\}$ because once they get close negative feedback dominates ($P_{\mathrm{FB}} < 0.5$).

Finally, we have a look at the resulting expected changes of the state variable s in one round: $\Delta s(s(t))$ (see Fig. 6.11). For the swarm model we get

$$\Delta s(s) = 4 \left(P_{\mathrm{FB}}(s) - \frac{1}{2} \right) \left(s - \frac{1}{2} \right).$$

For the Ehrenfest model we have one fixed point $s^* = 0.5$. For the Eigen model we have two fixed points: $s_1^* = 0$ and $s_2^* = 1$. For the swarm model we have two fixed points: $s_1^* \approx 0.23$ and $s_2^* \approx 0.77$. Between $s \approx 0.23$ and $s \approx 0.77$ the swarm is

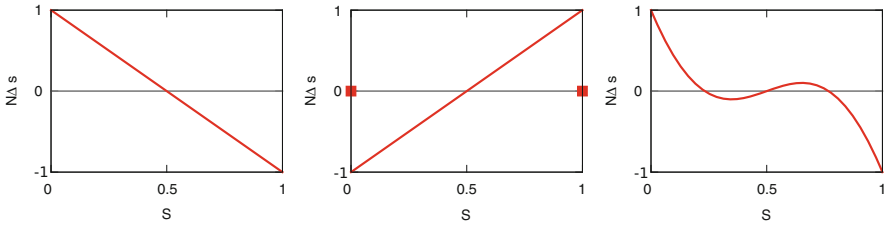


Fig. 6.11 Expected changes of the state variable s in one round: $\Delta s(s(t))$, for the (**a**) Ehrenfest, (**b**) Eigen, and (**c**) swarm model

driven away from $s = 0.5$ by positive feedback. Between $s \approx 0.23$ and $s = 0.5$ the swarm is driven towards $s = 0$. Between $s = 0.5$ and $s \approx 0.77$ the swarm is driven towards $s = 1$. Between $s = 0$ and $s \approx 0.23$ the swarm is driven towards $s = 0.5$ as well as between $s \approx 0.77$ and $s = 1$ the swarm is driven towards $s = 0.5$.

With the swarm urn model we have now a simple macroscopic model of collective decision-making that has some potential to represent even microscopic aspects, such as one robot persuading another robot. This urn model explicitly represents feedbacks and also helps to get a better understanding of positive and negative feedback as introduced in Sect. 1.3.1. Despite its simplicity, the swarm urn model has potential to be applicable to scenarios of swarm robotics.

6.5.2 Voter Model

The voter model [73] is a simple model of collective decision-making that has amazing properties. A robot i considers its neighbors' current opinions o_j with $j \in \mathcal{N}_i$. It picks one neighbor j at random and switches its own opinion to the opinion of that neighbor (if the robot's previous opinion is identical with the opinion of the picked neighbor, then it just keeps its opinion). Now you may ask yourself what kind of pointless behavior that is, picking opinions of random neighbors. Certainly, you don't want to adopt that as your approach to the next elections. However, the purpose of this model is not just to give a baseline of the simplest behavior, instead it turns out that the voter model is competitive. The voter model implements a slow decision-making process but it is very accurate, that is, it helps the swarm to converge on the correct decision [395]. This high degree of accuracy based on randomly picking neighbors may be counterintuitive but it can be understood in the way that it is slow while not paying much attention to the current state of the system. If there is a majority for a certain opinion, then it is more likely to be picked. However, there is no determined process that would enforce the win of a local majority. Hence, the decision process is rather robust.

6.5.3 Majority Rule

The majority rule is another simple model of collective decision-making. In contrast to the voter model, it seems more intuitive. A robot i checks its neighborhood group $\mathcal{G}_i$ and counts the occurrence w_j of each option O_j. The robot then switches its opinion to the most frequent option O_k with $k = \text{argmax}\, w_j$, that is, the majority within its neighborhood group. Compared to the voter model the majority rule is less accurate but faster [395]. Intuitively, this can be understood in the following way. Say, we have an extreme situation where each robot has an atypical sensor range that covers the complete swarm. Say, there are only two options O_1 and O_2 and initially each robot has a 50–50 chance of having one of them as opinion. Then the overall distribution of the two options in the swarm is also close to a 50–50 situation. Hence, the initial majority is determined randomly but all robots would update their opinion and follow that randomly determined initial opinion. In 50% of the cases, this initial majority will be the worse option of the two. With the voter model, even a robot with this extreme long sensor range would only pick randomly one of the robots out of the swarm and switch to its opinion. If dissemination is correlated to the quality of an option, the swarm would still have a chance to make a correct collective decision.

6.5.4 Hegselmann and Krause

A popular model across many different fields of research is the Hegselmann–Krause model [183]. Considering its simplicity, it is rather surprising that it was published as late as 2002. The robots choose from a continuum of possible options x defined on an interval $x \in [0, L]$ with $L \geq 1$. So the robots literally position themselves, although the interval $[0, L]$ needs not to be directly associated with positions in real-world space. The idea is that each robot moves to the mass center of its neighbor group. The update rule in each time step for a robot i is given by

$$x_i = \frac{1}{|\mathcal{G}_i|} \sum_{j \in \mathcal{G}_i} x_j, \tag{6.6}$$

where $\mathcal{G}_i = \{1 \leq j \leq N : ||x_i - x_j|| \leq 1\}$. Furthermore, one assumes that all agents simultaneously update their position, which does not mean that it is illegitimate to define and investigate an asynchronous Hegselmann–Krause variant. The robots are greedily following this model. They quickly gather with their neighbors but don't explore whether there are more peers just behind their horizon. As a result one gets many small clusters of aggregated robots for reasonable values of L, swarm size N, and initial distributions of opinions (see left part of Fig. 6.12). To get a more useful model for swarm robotics one can add a noise term ε_i [102], that can, for example be sampled from a random uniform distribution on the interval $\varepsilon_i \in [-0.5, 0.5]$ yielding

$$x_i = \frac{1}{|\mathcal{G}_i|} \sum_{j \in \mathcal{G}_i} x_j + \varepsilon_i. \tag{6.7}$$

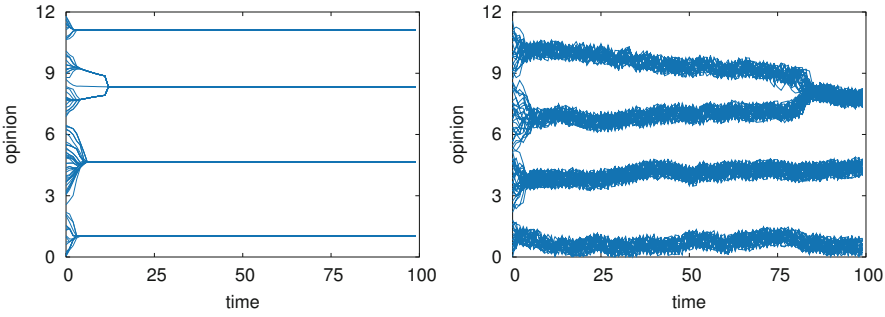


Fig. 6.12 Examples of trajectories in "opinion space" for (left) the original Hegselmann–Krause model, Eq. (6.6) and (right) the variant with exploration, Eq. (6.7); used parameter setting: $L = 12$, swarm size $N = 100$, initial robot positions sampled from a random uniform distribution on $[0, 12]$

The noise term ε_i then implements explorative behavior that enables robots to (temporarily) move out of these small clusters and eventually join other clusters (see right part of Fig. 6.12).

6.5.5 Kuramoto Model

The Kuramoto model is usually used to describe synchronization processes by coupled oscillators [225]. A robot is represented by an oscillator (e.g., a harmonic oscillator from classical mechanics or even simpler) and robots "communicate" by being coupled to other robots. Each robot $i \in \{1, 2, \ldots, N\}$ has as an internal state its current phase $\theta_i \in [0, 2\pi]$. In the standard Kuramoto model, the phase is updated by

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{N} \sum_{j=1}^{N} \sin(\theta_j - \theta_i), \tag{6.8}$$

for the robot's preferred frequency ω_i and coupling strength K. The standard model, however, has global coupling, that is, each robot talks to all other robots. Instead we prefer local coupling

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{|\mathscr{G}_i|} \sum_{j \in \mathscr{G}_i} \sin(\theta_j - \theta_i), \tag{6.9}$$

with $\mathscr{G}_i = \{1 \leq j \leq N : \mathrm{dist}(\theta_i, \theta_j) \leq r\}$, for some (sensor) range r, and $\mathrm{dist}(\cdot)$ respects the torus-like concept of radians

$$\mathrm{dist}(x, y) = \min(\|x - y\|, 2\pi - \|x - y\|). \tag{6.10}$$

Initially all robots start with random phase. The coupling introduces a feedback pushing towards synchronization. So this is similar to a popular experiment with metronomes (usually an annoying tool to make music students keep the pace). For example, one can put five asynchronous metronomes on a board and the board on two beverage cans. The board then starts to move slight back and forth, imposing a force on all metronomes, basically a global coupling, that synchronizes them. With better synchronization the board moves even more and imposes a bigger force on the metronomes (positive feedback). In the end, all metronomes are synchronized.

Many variants of the Kuramoto model are possible and have been proposed. One can weight the coupling by the distance, one can add noise, one can impose a structure in the form of a 1-d or 2-d array of oscillators, etc.

A beautiful work uses the concept of coupled oscillators to model a phenomenon that was observed on the Millennium Bridge in London [36, 367]. Due to initial fluctuations the bridge started to move laterally back and forth. People walking on the bridge react to that motion by unconsciously synchronizing their gait. With a crowd size above a critical threshold, the motion of the bridge is reinforced (positive feedback) and starts to move even more. This is a self-organizing system and a nice example of a swarm effect. It was found that the phenomenon is only observed once the swarm size is more than a critical value of $N_c = 160$ in the case of the Millennium Bridge. The work by O'Keeffe et al. [297] is also related to this as it allows oscillators to move and swarm while synchronizing. An example of an application of the Kuramoto model in robotics is given by Moioli et al. [283]. They have used the model as a benchmark.

6.5.6 Axelrod Model

Axelrod's model of the dissemination of culture [21] is an abstraction of cultural diffusion. We have an $L \times L$ square lattice of cells. In each cell there lives an agent with a culture that is characterized by a list of n features (integers on a defined interval). All agents are initialized to a random culture. The update in each time step is done in the following way. First, an agent i is randomly selected. Second, a neighbor j of agent i is selected randomly. Agent i and j interact with each other with probability $P = s/n$, where s is the number of cultural features that are the same for agent i and j. Agent i selects a cultural feature of agent j that differs from its own and overwrites its own value with that of agent j. This process is continued until either all neighboring agents share the same culture or have completely different cultures (i.e., $P = 0/n = 0$). The main message of Axelrod's model is that "local convergence can lead to global polarization"and that "simple mechanisms of change can give counterintuitive results" (large territories with little polarization) [21].

6.5.7 Ising Model

Our starting point is the well-known Ising model that was originally developed as a toy model to investigate ferromagnetism and phase transitions [429]. The Ising model is popular across fields and physicists make fun of those applying it to alternative systems, quoting from Sznajd-Weron and Sznajd [371]:

> The Ising spin system is undoubtedly one of the most frequently used models of statistical mechanics. Recently, this model has also become the most popular physics export article to "other branches of science" such as biology, economy or sociology. There are two main reasons for that: first verbalized by Nobel prize winner Peter. B. Medawar – "physics envy" – a syndrome appearing in some researchers who would like to have such beautiful and relatively simple models as physicists have (for example the Ising model).

The original Ising model is defined on a lattice, that is, we have points positioned regularly and, for example, connected to four neighbors each (square lattice, see Fig. 6.13). Usually one assumes that this lattice is static, that is, each point keeps its neighbors for ever. In addition to the lattice, each of these points has a so-called spin s which is binary and can only take one of two values: $+1$ or -1. The standard definition of the Ising model is done via its Hamiltonian H which gives the overall energy of the system. We have

$$H = -J \sum_{\langle ij \rangle} s_i s_j - B \sum_i s_i, \tag{6.11}$$

for a coupling constant J, an external magnetic field with constant B, the term $\langle ij \rangle$ gives all pairs of neighboring spins as defined by the lattice, and we have N spins, hence $i \in \{1, 2, \ldots, N\}$. Here, we do not need an external field and set $B = 0$ which yields

$$H = -J \sum_{\langle ij \rangle} s_i s_j. \tag{6.12}$$

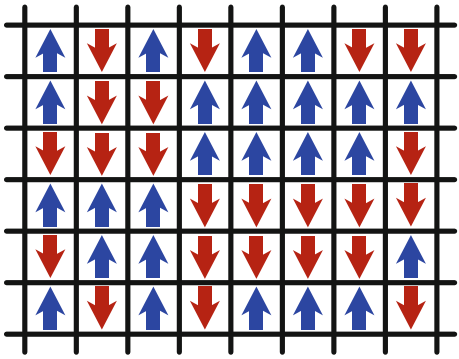


Fig. 6.13 Example configuration of a 2-d Ising model. Red arrows represent spin $s = +1$ and blue arrows represent spin $s = -1$

Interestingly the Hamiltonian does not describe the local behavior of individual spins; instead, the physicist is initially satisfied by knowing the global system state only. The cultural clash here is that swarm roboticists have to design robot behaviors whereas physicists find readily engineered systems that just need to be modeled. The actual physical system described by the Ising model (i.e., ferromagnetic systems) changes its spins spontaneously based on fluctuations and local interactions. Hence, the behavior of spins is not far from the behavior of our robots that also explore their environment by random actions and cooperate with neighboring robots.

Next, we define the behavior of the spins via a Monte Carlo simulation (Markov chain Monte Carlo simulations [39]). While physicists use that simulation only to calculate certain properties of the spin system and do not want to describe the actual behavior of spins by it, we take the local rules of the simulation as a definition of the agent behavior. We use the numerical method of the Metropolis algorithm as our definition of the agent behavior.

The Metropolis algorithm defines a so-called single-spin-flip dynamics and tries to flip a single spin per time step that implements the transition from a configuration v to the next configuration w. For that we choose a random site i with homogeneous selection probability $g(i) = 1/N$ and calculate the change in energy $\Delta E = E_w - E_v$ that a considered flip would cause. Then we accept the flip: (i) if $\Delta E \leq 0$, with a probability of one; and (ii) if $\Delta E > 0$, only with a defined acceptance probability A depending on ΔE defined by

$$A(v, w) = \exp(-\beta(E_w - E_v)) = \exp(-\beta \Delta E), \tag{6.13}$$

with $\beta = (kT)^{-1}$, for Boltzmann constant k and temperature T. For simplicity and because we do not want to simulate actual physical systems, we usually set $k = 1$. A negative energy difference ($\Delta E < 0$) corresponds to a majority decision of agents and a positive energy difference ($\Delta E > 0$) corresponds to a minority decision. For $T > 0$ we have a non-zero probability of doing a minority decision and for $T = 0$ minority decisions have zero probability. The temperature defines whether there is free energy available to do explorative actions, that is, actions that do not decrease the overall energy.

An interesting setup could be the following. We initialize with $T = \infty$ which gives us a random uncorrelated initialization of spins. Then we set $T = 0$, that is we cool the system, because we say that we want a homogeneous system state. As an effect and according to Eq. (6.13) we do not accept spin-flips that would increase the energy (i.e., a minority decision). Hence, we have a system that greedily optimizes locally. The interested reader can continue from here with a paper by Galam [131] that connects an Ising model directly with group decision-making.

6.5.8 *Fiber Bundle Model*

The fiber bundle model goes back to Peires [305]. A modern overview is, for example, given by Raischel et al. [321]. We assume a material that has N parallel fibers in a regular lattice. We investigate the reaction of the bundle set to a fore that pulls these fibers. Each fiber i is elastic and stretches until it breaks at an individual breaking threshold θ_i. Each fiber's individual threshold θ_i is an independent identically distributed random variable. Choosing a distribution is a modeling decision. A typical choice is a Weibull distribution

$$P(\theta) = 1 - \exp\left(-\left(\frac{\theta}{\lambda}\right)^m\right), \tag{6.14}$$

where m is the Weibull index and λ is the scale parameter. Once a particular fiber has failed, its load has to be shared by the remaining intact fibers. We distinguish two ways of sharing the load. In global load sharing the load is equally redistributed over all intact fibers. In local load sharing the entire load of the failed fiber is redistributed equally over its local neighborhood. For example, we can choose a von Neumann neighborhood and distribute the load among the four (not yet broken) neighbors (north, south, east, west). An interesting effect observed in this model is the occurrence of avalanches of breakings. We can count how many fibers break in an event of such an avalanche, which defines the avalanche size Δ. The distribution D of avalanches of size Δ follows a power law distribution with an exponent $5/2$ [321]

$$D(\Delta) \propto \Delta^{-5/2}. \tag{6.15}$$

In the context of swarm robotics, we can interpret the fiber bundle model as a time model for collective-decision. Say the task is not to choose from several options but to decide when to take a single option (e.g., escape or when to start construction etc.). Then "breaking" would be re-interpreted as a positive event of a robot's individual decision of wanting to take or taking action now. That decision then possibly triggers decisions in its neighborhood. The individual breaking threshold θ_i would correspond to the robot's individual sensitivity. An avalanche would correspond to the rapid propagation of a decision through the swarm. The power law distribution of avalanche sizes Δ would give us a hint of how many robots we should expect to take decisions per time.

A drawback of the fiber bundle model is that fibers are stationary and their neighborhoods do not change. The breaking threshold θ_i introduces an interesting heterogeneity that may have positive effects on the robot swarm but this relation is not clear.

6.5.9 Sznajd Model

The Sznajd model is also known as USDF model ("united we stand divided we fall"). An individual i has an opinion $S_i = -1$ for no or $S_i = 1$ for yes and an individual has two neighbors. In each time step two individuals S_i and S_{i+1} are randomly chosen. The opinion of their neighbors S_{i-1} and S_{i+2} is then updated following these two rules

- if $S_i S_{i+1} = 1$ then $S_{i-1} = S_i$ and $S_{i+2} = S_i$,
- if $S_i S_{i+1} = -1$ then $S_{i-1} = S_{i+1}$ and $S_{i+2} = S_i$.

This system always ends up either in a consensus state or stalemate state (disagreement). A power law for the decision time distribution with an exponent of -1.5 was found.

6.5.10 Bass Diffusion Model

The Bass diffusion model [31] is an abstraction of how an innovative product is adopted by consumers over time after its first release. The underlying assumption is that news about the product spread in a word-of-mouth fashion from past adopters to not-yet-adopters. An adoption of a product is the first-time purchase of that product. This is related to Arthur's concept of "lock-ins" [15] that occur, for example, when there is a "format war" between mutually incompatible proprietary formats (e.g., the videotape format competition between Betamax and VHS). However, here with the Bass diffusion model we only focus on one product and how it spreads throughout the market. The word-of-mouth process with adopters (consciously or unconsciously) persuading not-yet-adopters can be interpreted as a collective decision-making process with two options: to buy or not to buy. With the Bass diffusion model we can study the process over time.

The fraction of the market that adopts the new product at time t is $f(t)$ and the fraction of the market that has adopted the new product already at time t is $F(t)$. "The portion of the potential market that adopts at [time] t given that they have not yet adopted is equal to a linear function of previous adopters."[1] We have

$$\frac{f(t)}{1 - F(t)} = p + qF(t), \tag{6.16}$$

where p is the coefficient of innovation, q is the coefficient of imitation. For a (potential) market of size N we get sales $S(t) = Nf(t)$, which can be expressed depending on p, q, and N only [31].

[1] http://www.bassbasement.org/bassmodel/bassmath.aspx.

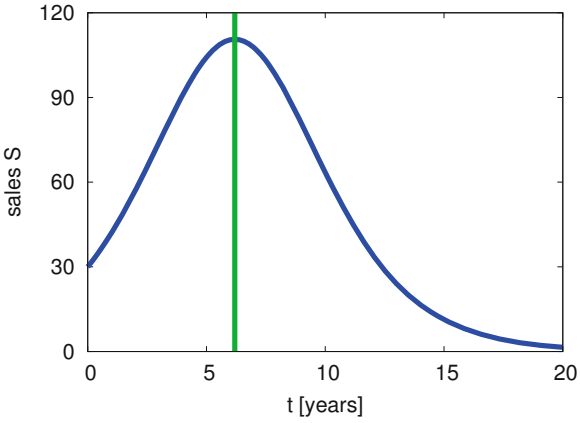


Fig. 6.14 Sales in the bass diffusion model, market size $N = 1000$, coefficient of innovation $p = 0.03$, coefficient of imitation $q = 0.38$, time of peak sales $t^* \approx 6.19$

$$S(t) = N\frac{(p+q)^2}{p}\frac{e^{-(p+q)t}}{\left(1+\frac{q}{p}e^{-(p+q)t}\right)^2}. \tag{6.17}$$

The time of peak sales t^* is given by $t^* = \frac{\ln q - \ln p}{p+q}$. An example plot of sales over years is shown in Fig. 6.14. The market size N can be interpreted as swarm size and $S(t)$ specifies how many robots have been convinced about option "buy" at time t.

6.5.11 Sociophysics and Contrarians

Galam [133] calls the set of models that he developed over several decades "sociophysics." He focuses on collective decision-making, democratic voting, and opinion dynamics. Out of the many works of his [132–136], we select a model of collective decision-making that incorporates so-called contrarians [132, 133]. "A contrarian is someone who deliberately decides to oppose the prevailing choice of others whatever this choice is" [132]. In a robot swarm, a contrarian could be a faulty robot or a sabotaging intruder. It could also be part of a second robot group deliberately added to the swarm (hence making it a heterogeneous swarm) to keep the swarm from converging on a consensus.

The idea of the contrarian model is to assume a certain fraction $0 \leq a \leq 1$ of contrarians in a population. The system state is given by $0 \leq s \leq 1$, that is the fraction of agents in favor of option O_1 for a binary decision problem $O = \{O_1, O_2\}$. Depending on the current system state s_t the contrarians are always against the current majority. We want to investigate the evolution of this system, that is, a time series $s_0, s_1, s_2, \ldots$. The process of how an agent switches opinion is based on a simple assumption. We say that in each time step arbitrary groups $\mathscr{G}$ of agents form. These groups have a defined constant size of $|\mathscr{G}| = r$. Any possible

spatial correlations between agents and their opinions are ignored. We also assume that within these groups a majority rule determines the opinion switch. Now we are interested in the resulting update rule for the system state s. We start without contrarians, that is, $a = 0$. We have to do the combinatorics and look through all possible combinations of groups of size r with two opinions o_1 and o_2. We sum over all combinations that result in a majority decision for option o_1 and get the update rule

$$s_{t+1} = \sum_{m=\frac{r+1}{2}}^{r} \binom{r}{m} s_t^m (1 - s_t)^{r-m}. \tag{6.18}$$

To add the influence of contrarians, we just have to include a basically inverse term. It accounts for those groups where a majority would have emerged for opinions o_2 but due to contrarians was overruled. The two terms are then weighted by the fraction of contrarians a. We get

$$\begin{aligned} s_{t+1} =& (1 - a) \sum_{m=\frac{r+1}{2}}^{r} \binom{r}{m} s_t^m (1 - s_t)^{r-m} \\ &+ a \sum_{m=\frac{r+1}{2}}^{r} \binom{r}{m} (1 - s_t)^m s_t^{r-m}. \end{aligned} \tag{6.19}$$

For example, with a group size $r = 3$ we get

$$\begin{aligned} s_{t+1} =& (1 - a)(s_t^3 + 3s_t^2(1 - s_t)) \\ &+ a((1 - s_t)^3 + 3(1 - s_t)^2 s_t). \end{aligned} \tag{6.20}$$

The resulting functions $s_{t+1}(s_t)$ for varied $a \in \{0, 0.05, 0.1, 0.25\}$ are given in Fig. 6.15. The crossings of function $s_{t+1}(s_t)$ with the diagonal $s_{t+1} = s_t$ are the fixed points. Depending on the contrarian fraction a there is a qualitative change. For $a \in \{0, 0.05, 0.1\}$ the values $s_1^* \approx 0$ and $s_2^* \approx 1$ are stable fixed points while $s_3^* = 0.5$ is an unstable fixed point. However, for $a = 0.25$ there is only one stable fixed point $s_1^* = 0.5$, which means the swarm converges on the undecided state of 50% in favor of opinion o_1 and 50% in favor of opinion o_2. A relatively small fraction of contrarians is enough to obstruct the decision-making process. In addition, the steepness of the function $s_{t+1}(s_t)$ determines how much time the swarm requires to converge (steeper is faster). A small fraction of contrarians $a = 0.05$ already slows down the process.

As mentioned above the use of contrarians in robot swarms can be to avoid consensus. In a dynamic environment it is desirable to have always a small percentage of robots that have a different opinion than the majority. For example, these robots keep visiting and exploring areas corresponding to their opinion and

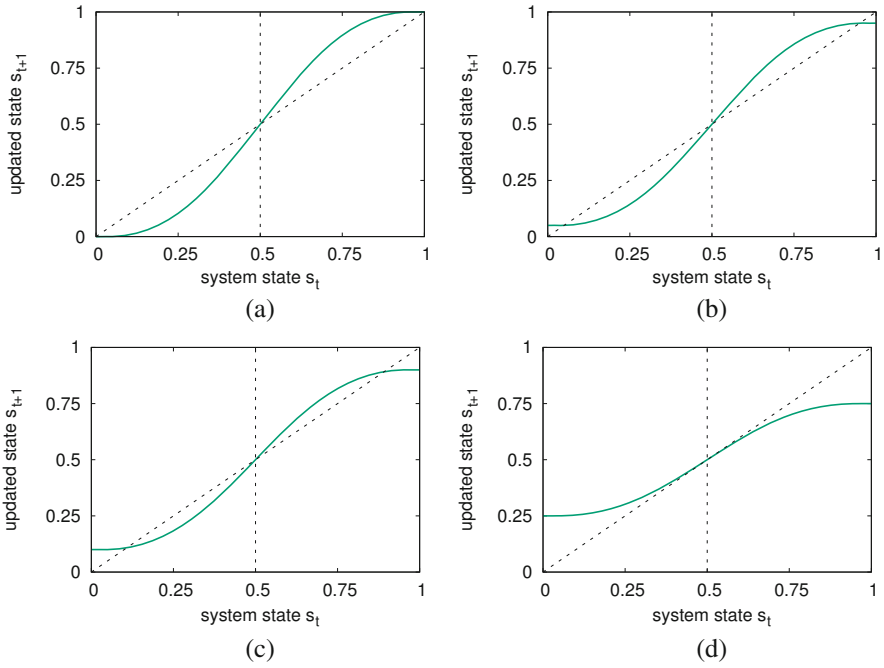


Fig. 6.15 Sociophysics and contrarians [132], change of system states s (fraction of agents in favor of opinion o_1) depending on the fraction of contrarians a in the swarm, group size $r = 5$, Eq. (6.19). (**a**) Contrarians $a = 0$ (**b**) Contrarians $a = 0.05$ (**c**) Contrarians $a = 0.1$ (**d**) Contrarians $a = 0.25$

would hence detect a dynamic increase in quality. An adaptive swarm has to avoid consensus or implement exploration in an alternative way. The concept of contrarians is similar to what Merkle et al. [272] call "anti-agents." In an aggregation scenario, for example, anti-agents try to prevent clustering. Merkle et al. [272] suggest to use anti-agents to prevent undesired emergent effects in a swarm.

6.6 Implementations

6.6.1 Decision-Making with 100 Robots

Only since recently the robot swarms get considerably bigger. Since 2014, reported hardware experiments in swarm robotics go up to 1000 robots [332]. Here we have a look into a swarm of 100 Kilobots [331] that are supposed to explore two potential target areas and collectively decide, which one the swarm should choose. In a series of studies Valentini et al. have investigated the impact of the voter model and the majority rule on collective decision-making in robot swarms [395–398].

The task for the robot swarm is to choose between the blue site and the red site (see Fig. 6.16). Initially the robots are positioned within the middle of the arena, they are initialized to a blue or red opinion with equal probabilities, and they know nothing about how desirable these areas are. The robots know, however, how to

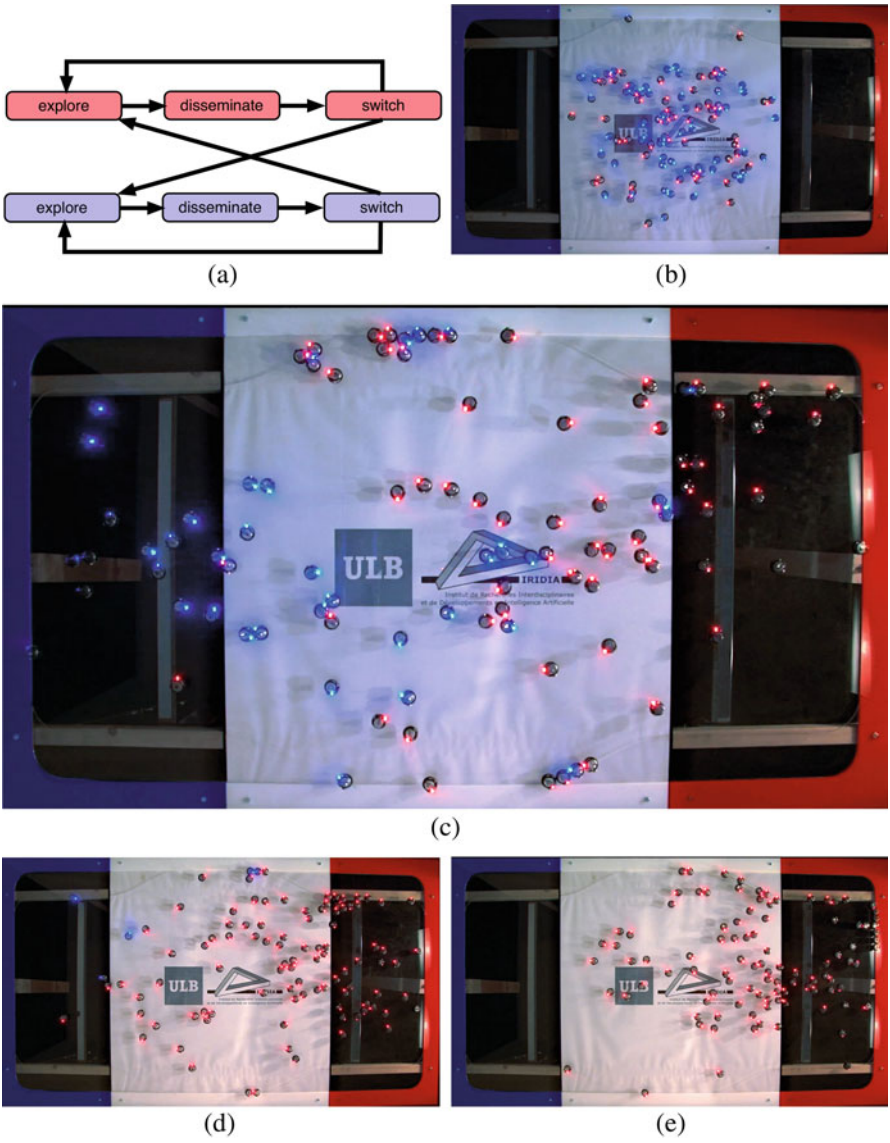


Fig. 6.16 Collective decision-making with 100 Kilobots [395]. (**a**) Finite state machine of robot behavior (**b**) Initial configuration of robot experiment (**c**) Early configuration of robot experiment (**d**) Intermediate configuration (**e**) Late configuration

navigate to them. Behind the red site, there is a light beacon (if you look closely you can see the respective shadows of the robots in Fig. 6.16c). The robots know that when they approach the light beacon (phototaxis) then they navigate towards and eventually onto the red site and if they approach areas of lower light intensity (anti-phototaxis) then they navigate towards the blue site. Once the robots are on a site they can explore it. In this experiment that is emulated by IR messages that can only be received by robots on that site. The message contains values of the actual site quality. Here, the red site has a higher quality than the blue site, hence, we expect a consensus on red.

The robots follow the procedure of collective decision-making in three phases as discussed above (Fig. 6.5). The robots are controlled by a probabilistic finite state machine (Fig. 6.16a) that has an exploration, dissemination, and switching state for both opinions (red and blue). In the exploration state the robots navigate to the respective site of their current opinion, explore its quality, and navigate back into the white area in the middle. During the following dissemination state the robots move randomly within the white area and send messages containing their current opinion as local broadcasts into their neighborhood, that is, the area around them that is reached by their infrared signal (a few centimeters, whereas the overall arena is of size 100×190 cm). Finally, in the switch state the robots move randomly within the white area, listen for messages of their neighbors for a while, count the number of messages of either of the two opinions, and then switch. Two switching rules were investigated. Using the voter model the robot picks a random message and switches to the opinion disseminated in that message. Using the majority rule the robot compares the count of "blue messages" to "red messages" and switches to the majority.

Independent of whether the voter model or majority rule is used, based on the description given so far, there would be no bias towards any of the two sites because the measured site quality has no influence yet. This bias is implemented by varying the time spent in the dissemination state depending on the measured quality of the site. A higher quality results in more dissemination time. The mean time is directly proportional to the quality. This is called modulated positive feedback because the robots themselves influence continuously the degree of positive feedback.

An experiment with these 100 robots takes 90 min, which may seem quite long. The Kilobots cannot move fast with their little vibrating legs. Also the exploration phase was emulated to have a certain minimal duration although the robots could perform it much faster technically. As seen in Fig. 6.16 and as reported by Valentini et al. [395], the approach is effective. In the end, the swarm reaches consensus on red. This scenario could be an initial phase, for example, of swarm construction. Initially the swarm has to agree on where to start to build in order to focus all power of the swarm on one building site.

The difference between the voter model and the majority rule is that the voter model turns out to be more accurate (i.e., the correct consensus is found more frequently), while the use of a majority rule results in a faster decision-making process [395]. This is in good correspondence to respective findings in ants [126] and for decision-making of humans [46]. It is known as the speed-vs-accuracy

tradeoff and seems to basically be a natural law of decision-making. Either you are fast but then you cannot be accurate or you are accurate but then you cannot be fast. Most of us can confirm that rule from our own experience. Certainly the opposite is not necessarily true. Spending a lot of time on decision-making does not always improve the quality of the final decision. When designing a robot controller for collective decision-making in a robot swarm, one has to decide whether accuracy or speed is more relevant for the respective application.

6.6.2 Collective Perception as Decision-Making

Although Valentini et al. [393] call their approach "collective perception," it can also be seen as a collective decision-making scenario. The exploration phase is of even more interest here than in the above example with the Kilobots. The robots have to decide about a global feature of their environment. They move on a floor with white and black tiles (see Fig. 6.17) and they have a ground sensor to detect the brightness. The task for the swarm is to determine whether black or white tiles are in majority. In contrast to the above example, the robots have not to navigate to a certain area depending on their opinion, instead they can permanently explore. The frequency of black and white tiles is a global feature of the environment. The task difficulty can be regulated by choosing either a clear majority of one of the tile types (e.g., 66% black as in Fig. 6.17a) or a relation that is closer to the 50–50-state (e.g., 52% black as in Fig. 6.17a).

In this experiment 20 e-puck robots [285] were used. The robots do random walks and measure the time for how long their ground sensor detected the tiles of their current opinion (e.g., in the case of opinion "black," the robot measures the time for how long its ground sensor detected black tiles). Again, the dissemination time is then modulated according to that measurement, which serves as an option quality here. Valentini et al. [393] tested again the voter model and the majority rule. In addition, also a method called "direct comparison" was used that would just directly compare the quality measures to determine whether the robot should switch its opinion. All three approaches are effective. The overall results are mixed. The majority rule results in fast decisions as expected. The direct-comparison method works well in rather small swarms but does not scale well with increasing swarm size and also has difficulties with harder tasks. The voter model resides arguably between the two other approaches but the results are not totally clear [393].

The unique aspect of this scenario is that collective decision-making is combined with collective perception. Similarly to what possibly happens in our brain when we look at Fig. 6.17, the robots have to collectively estimate the majority. The reported approach [393] obviously tries to keep the communication between robots simple, but more information could be shared. For example, robots could calculate averages over the measured qualities. The random walk could be replaced by a dispersion approach while still allowing the robots to mix. That way the robots may cover more area and may gather more representative samples.

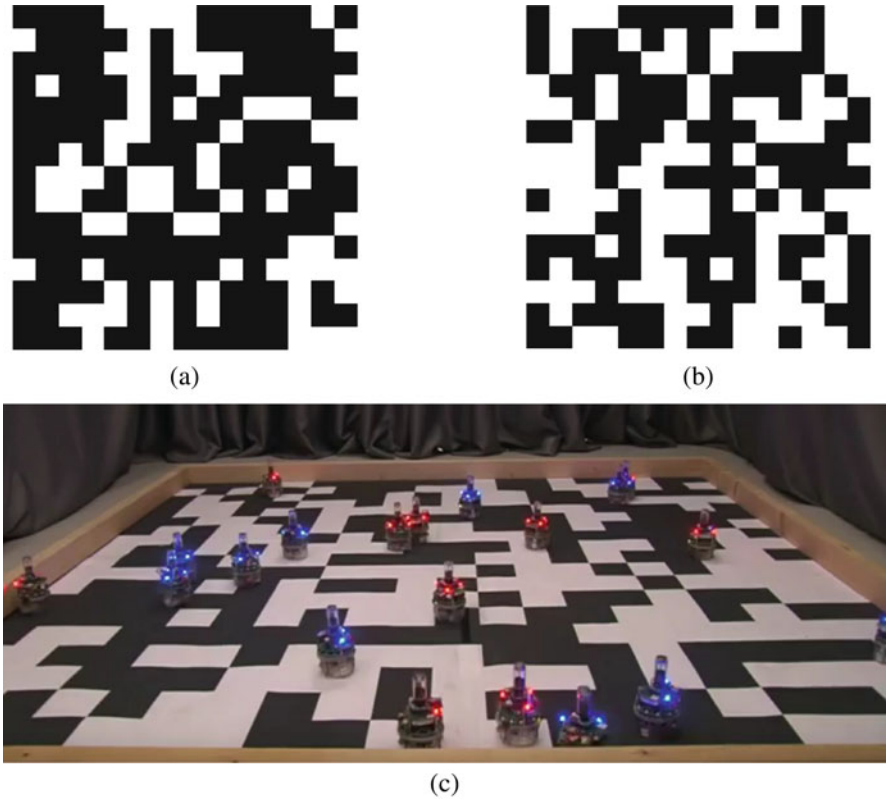


Fig. 6.17 Collective perception as collective decision-making, the robots have to determine whether black or white tiles are in majority; two environments: easy with 66% black and hard with 52% black [393]. (**a**) Easy (66% black) (**b**) Hard (52% black) (**c**) Robot experiment with 20 e-puck robots

6.6.3 Aggregation as Implicit Decision-Making

We can interpret aggregation behaviors in robot swarms as collective decision-making. At least implicitly the swarm agrees on an area to meet. The number of options is potentially infinite because space is continuous. The decision-making aspect gets more clear, once the robots have to select from a given set of possible aggregation areas. If all of these areas have the same utility or if at least the best ones are of equal utility, then the swarm faces a symmetry-breaking problem. The robot swarm has to find a consensus on which of the equivalent areas it wants to choose. Even if all areas are of different utility, the decision-making can still be implemented by a collective approach.

A popular example of an aggregation algorithm for swarms is BEECLUST [17, 19, 211, 212, 344, 349]. It implements an adaptive aggregation behavior, that is,

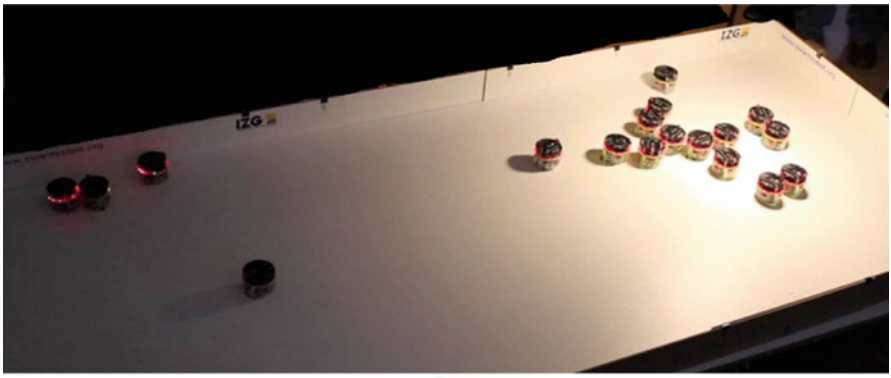
not all positions in space are of equal utility for aggregation. Instead a value can be measured at any point in space to determine this spot's utility. For example, this feature can be light, temperature, or a concentration of a chemical. Originally the algorithm was inspired by the aggregation behavior of young honeybees. A key concept of the algorithm is that it relies on infrequent measurements. Instead of implementing a gradient ascent, that is, a robot follows the spatial feature's slope, the spots of high utility are determined by social interactions of robots. Whenever two robots meet they measure the local feature and stay stopped for a period of time proportional to the measured feature. In a positive feedback process, clusters of aggregated robots form mostly close to areas of high utility. For examples of implementations with different robot platforms, see Fig. 6.18. The details of this approach are discussed later in Chap. 7.

By interpreting BEECLUST as a collective decision-making approach, we find several unique properties compared to a standard decision-making process as described above. Instead of having a few clearly defined options O, the robots can in principle choose from infinitely many positions in continuous space. However, by forming clusters they create virtual discrete options that can at least be distinguished by an external observer. If you like, aggregating robots mark positions in the "decision space" as options and try to attract other robots to their opinion.

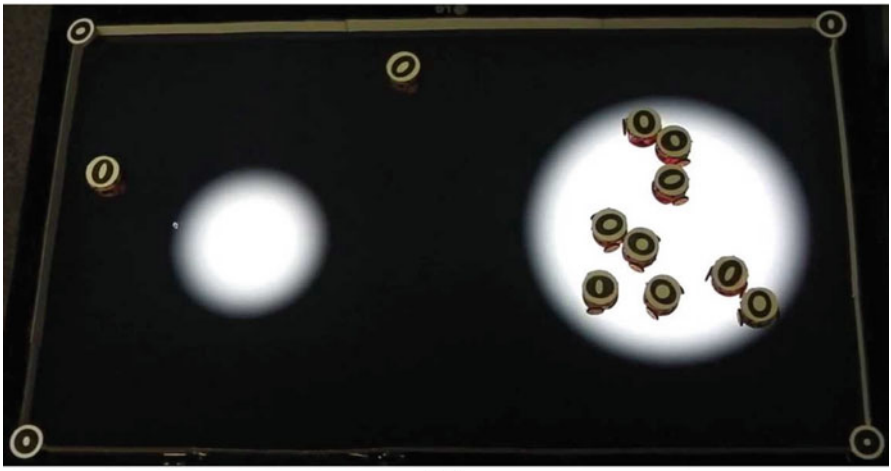
6.7 Further Reading

A long text that is fully dedicated to collective decision-making in swarm robotics is that of Valentini [392]. Especially, the introduction of the concepts is a great read and the overall approach is rather theoretical-oriented. A shorter and related text is that of Valentini et al. [394]. Out of the domain of swarm robotics but still focused on collective decision-making and opinion dynamics are the works of Serge Galam on his "sociophysics" [132–136]. A paper of similar style is that of Castellano et al. [66], which covers a wide range of models related to collective decision-making from the statistical physics perspective.

In addition, there are rather specialized but relevant highlights in the literature. For example, the work on discrimination of resources by Campo et al. [61]. Scheidler [340] offers a great theoretical analysis for the influence of latencies. Similarly, Montes de Oca et al. [87] investigate latencies in a swarm. With focus on animal behavior the work of Couzin et al. [76, 77] is highly relevant. Also Szopek et al. [372] report interesting behaviors in honeybees and Dussutour et al. [106] report the essential influence of noise in decision-making. An example of collective decision-making in a bio-hybrid system is given by Halloy et al. [163]. Influencing natural swarms with artificial swarms is an intriguing future option for pest control. Finally, Reina et al. [324] address software-engineering approaches for programming collectively deciding robot swarms. In a recent paper, Hasegawa et al. [180] study the connection between individual threshold-based yes/no–decisions of individual swarm members and the resulting macroscopic decision of the swarm.



(a)



(b)

(c)

Fig. 6.18 Implementations of the BEECLUST algorithm on different robot platforms. (**a**) e-pucks (Ralf Mayet and Thomas Schmickl) (**b**) Colias robots on a 42 inch (107 cm) LCD screen by Arvin et al. [18] (**c**) Thymio II by Wahby et al. [406]

6.8 Tasks

6.8.1 Aggregation at Specified Spot

We start from the solution of task Sect. 4.15.1 "Behaviors of robot swarms." There we have implemented a swarm behavior that aggregates the swarm at an unspecified spot. Now we want to aggregate the swarm at a specified spot, the brightest area.

Define a light distribution in space as, for example, shown above. Provide your robots with a light sensor. Now the time that the robots stay stopped should also be influenced by measurements of light. How can that be done? Implement that in your simulation and observe whether the swarm splits between bright areas.

6.8.2 Urn Model for Locust Scenario

We use the locust simulation again. We work with the following parameters: circumference $C = 0.5$, speed of 0.01, perception range of $r = 0.045$, swarm size of $N = 50$, spontaneous direction switch with a probability $P = 0.15$ per time step.

In your simulation you still have to keep track of the left-goers L_t at a time step t and count them. We are interested in the average change of L_t in dependence of itself. That is, we are searching for a function $\Delta L(L)$. To measure this function, we simulate the locusts for 100 time steps. These 100 time steps are needed by the locusts to organized themselves. We simulate additional 20 time steps while we keep track of L_t. At each time step $t \in [101, 120]$ we "store" the change of L within one time step $\Delta L = L_t - L_{t-1}$, for example, by adding it to an array. It's an array because we want to measure ΔL in dependence on L itself. Hence, you can have a variable *deltaSum*[] in your program and for each time step $t \in [101, 120]$ you do: $deltaSum[L_{t-1}] += deltaSum[L_{t-1}] + (L_t - L_{t-1})$. Also keep track of how often you have added something to an entry (e.g., a variable *count*[]) in order to normalize your measurements. Repeat these runs many times—up to 50,000 samples could actually be useful. Finally write your data of the measured function $\Delta L(L)$ to a file for later use.

(a) Plot the data for $\Delta L(L)$. What is the mathematical and domain-specific meaning of the positions L^* with $\Delta L(L^*) = 0$?
(b) Next we fit the swarm urn model to this data. Remember the equations

$$P_{\mathrm{FB}}(s, \varphi) = \varphi \sin(\pi s), \tag{6.21}$$

$$\Delta s(s) = 4 \left(P_{\mathrm{FB}}(s, \varphi) - \frac{1}{2} \right) \left(s - \frac{1}{2} \right), \tag{6.22}$$

whereas P_{FB} is the probability of positive feedback and $\Delta s(s)$ is the expected average change of the considered state variable. Here these are the left-goers and hence s is the ratio of left-goers $s = L/N$. In addition, we have to introduce a scaling constant c to the idealized equation, which yields:

$$\Delta s(s) = 4c \left(P_{\mathrm{FB}}(s, \varphi) - \frac{1}{2} \right) \left(s - \frac{1}{2} \right) \tag{6.23}$$

Fit this function to your data, that is, try to find appropriate values for the scaling constant c and the feedback intensity φ. Be sure to keep φ within a reasonable interval: $\varphi \in [0, 1]$.

c. Plot your data together with the fitted function. Also plot the resulting probability of positive feedback P_{FB}. What can be said about these results?

Chapter 7
Case Study: Adaptive Aggregation

"A smart machine will first consider which is more worth its while: to perform the given task or, instead, to figure some way out of it."

—Stanisław Lem, The Futurological Congress

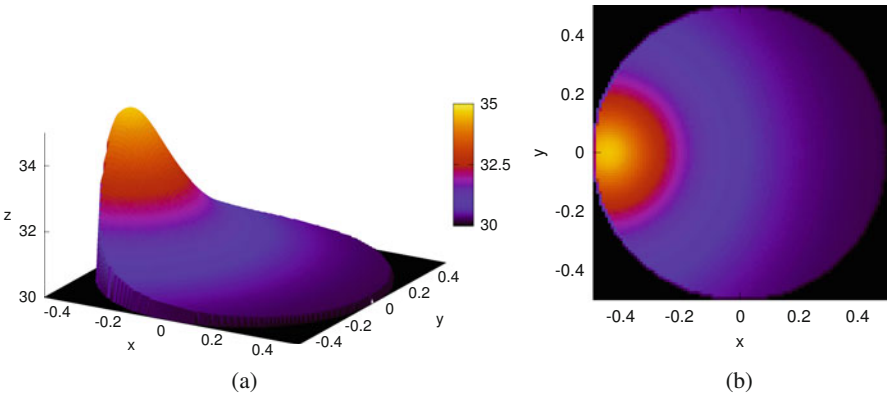
Abstract We try the impossible and summarize almost all we have learned about in this book.

In this little case study we try to integrate all different techniques that one requires to design a system of swarm robots. The task is rather simple and we focus on adaptive aggregation, that is, the swarm has to aggregate at a particular spot determined by environmental features. We follow a biological role model and apply standard modeling techniques of swarm robotics. Finally, the control algorithm and the micro-macro model are verified against robot experiments.

7.1 Use Case

Let's imagine we are several years into the future and meanwhile you have founded a successful software company that produces robot control software. Of course, you specialize in swarm robotics. By emulating some of the software-engineering speak, we say we go through a swarm robotics use case. Say, a customer orders software from you and specifies it in the following way. The software controls a robot swarm, which aggregates at a certain spot determined by sensor input and stays flexible to a dynamic environment. That sensory input could be anything but say it is temperature. Then we want aggregation at the warmest spot. Possible scenarios depend on the distribution of temperature, such as shown in Fig. 7.1. However, much more complex scenarios are possible. The temperature distribution could have local optima, many scattered spots instead of just one well-defined, smooth peak. There could also be systematic plateaus that give no information via the gradient (i.e., the spatial derivative is zero and we cannot guess in what direction we may find a peak).

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2_7



(a) (b)

Fig. 7.1 Spatial temperature distribution for aggregation at the warmest spot. (**a**) Side view (**b**) Top view

7.2 Alternative Solutions

You may want to guess an efficient approach immediately but we first go through a few alternative solutions.

7.2.1 Ad-hoc Approach

A simple ad-hoc approach may be to let robots drive randomly around while measuring. A robot could keep track of the biggest measured values. After some time, a robot stops when approaching a big value again. In this approach we would ignore the multi-robot setting. As an addon we could share the biggest measured values with neighboring robots whenever they pass by. This approach has the disadvantage that we make heavy use of the sensor (may cost additional energy). We also require a rather robust and reliable sensor, because a faulty sensor reading may give us a far-off maximum. Once the robot tries to find that maximum again, we run into a deadlock as the robot will not find it. Also note that the simplified version of this approach—just stopping once we find a relatively high value after some time—is directly related to the so-called secretary problem [129].

7.2.2 Gradient Ascent

A slightly more developed approach could be to implement a gradient ascent, that is, robots try to detect the spatial derivative of the temperature either at a given position or approximated between two nearby points. This can arguably be called the

standard engineering approach. We can include the multi-robot setup by allowing robots to communicate promising directions, that they have determined, to their neighbors. It comes with a number of requirements. First, the approach only works if the gradient is well-defined and the temperature distribution is rather smooth, that is, there are no or only few sharp peaks. For a temperature distribution that is maybe a fair assumption. However, second we require that there are no local optima. Otherwise robots get trapped at mediocre peaks. Third, we also require reliable sensors to detect the gradient. We either need two sensors to do two measurements at different positions at the same time or we need memory to measure over time by moving the robot in between. Especially the approach with two sensors is questionable. We need double the hardware if we choose to put two sensors on the robot. Also depending on the size of the swarm robot we may not be able to place the sensors at a sufficiently long distance to each other (e.g., consider the Kilobot's 33 mm diameter, see Sect. 2.5.4).

7.2.3 Positive Feedback

Maybe it is useful to design our approach as an inherently collective approach from the very start. In addition, we can also follow the definition of self-organization (see Sect. 1.3) and add an important ingredient: positive feedback. Putting both together we can get something like the following. Each robot searches for a good spot (e.g., a high temperature that it hasn't seen for a while) and once found the robot hangs around there for a while. If nobody joins the robot, it gives up, and tries to find another spot. This is collective by design and we have positive feedback because clusters of stopped robots may form and grow. A drawback, however, may be that robots spend too much time in globally uninteresting regions. Also, maybe too many robots may stay stopped lonely with low chances of ever seeing another robot, for example, if the swarm density is rather low (cf. stick pulling, see Sect. 4.6).

7.3 Biological Inspiration: Honeybees

In order to elaborate on the positive feedback idea, we check a biological system for inspiration. As it turns out there is an animal behavior that resembles our desired robot behavior nicely. Young honeybees happen to aggregate in clusters in the bee hive influenced by the temperature. Here, "young" means that their age is below 24 h, that is, these honeybee ladies just got out of their cocoon and left their honeycomb. They can neither fly nor sting—a fact that is very much welcomed by honeybee researchers. So they only walk within the bee hive. Young honeybees prefer a certain temperature, which helps them to dry their wings. They stay in the hive, which is usually totally dark and has regions with different temperatures. It was observed that young honeybees aggregate at spots of 36 °C [372]. These young

honeybees are known to show reactions to social stimuli, that is, they change their behavior if they meet other bees.

Biologists have designed an experiment to learn more about the social aspects of the young bee's behavior. The young honeybees are put into an area of controlled conditions. It is a round area covered with honeycombs, it is all dark, and has a carefully adjusted temperature distribution. There is an optimal temperature (36 °C) spot on the left-hand side. The lowest temperature on the right-hand side is only slightly colder and has 30 °C. The overall temperature distribution is relatively homogeneous and the induced gradient information is rather "flat," that is, it is supposed to be difficult for a bee to determine a direction of increasing temperature [209, 372].

The experiment is monitored and data is recorded with an infrared camera and an array of temperature sensors in the floor below the honeycombs. The young honeybees are initially placed in the center of the area and then left for about up to 2 h. The experiment is repeated several times with either a single bee, a few bees, or many bees. The fascinating result is that a single bee cannot find the optimum spot. Instead she keeps walking around as shown in Fig. 7.2 (computer simulation). A few bees are inefficient but able to find the optimum. A big group of bees finds the optimum efficiently. We have a true swarm effect because we observe a significantly better individual performance with increasing group size (cf. Sect. 1.2.1). As a result we can catch the hypothesis that some social interaction is a crucial ingredient of this behavior.

If we want to find out about this missing link—what social interaction between bees helps them in finding the optimum?—then we have to look deeper into the biological approach. A challenge is that there is no known reductionist approach for determining an animal's behavior. There is no way of learning much from cutting

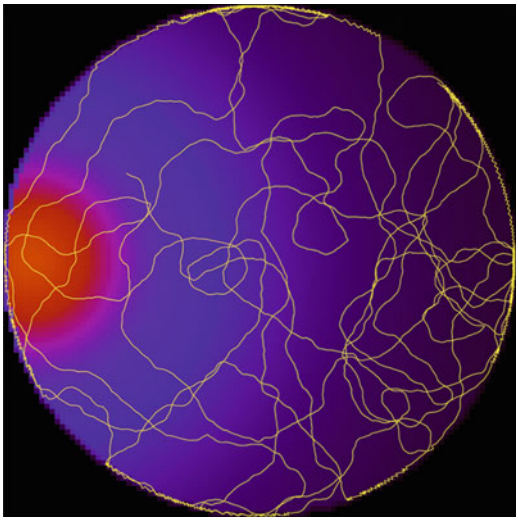


Fig. 7.2 Example trajectory of a young honeybee navigating in a circular area with a "flat" temperature distribution between 36 °C (left) and 30 °C (right). Shown trajectory is a computer simulation

the bee brain into pieces. There is, of course, neuroscience but it cannot give us the underlying algorithm of honeybee behavior. Instead biologists define a behavioral model as a hypothesis. For our little experiment here the behavioral model can look like this [349]:

step 1 move straightforward
step 2 obstacles or robots around?

(**a**) in case of an obstacle: turn away, return to step 1
(**b**) in case of a robot: stop, measure temperature, wait for some time (the longer the higher the temperature), U-turn, return to step 1

This is actually the BEECLUST algorithm [212, 344, 349] as discussed in Sect. 4.1. With this approach robots stay stopped longer in warmer areas, hence, increasing the chance of creating a growing cluster of robots once other robots join. We get a positive feedback effect (see Sect. 1.3) because bigger clusters are more likely to be found by robots. In addition, once a cluster has reached a certain size it is possible that robots get trapped within a cluster. These robots are physically blocked; even if they want to leave, they cannot. We notice that the positive feedback depends on two features, one feature that we can control ourselves (waiting time) and another feature we cannot control because it is a natural effect (physically blocked robots). The above behavioral model ignores that there could be areas that are too hot for the bees (or our robots). In a variation we could wait for shorter times if the measured temperature is very high.

7.4 Model

Next we want to model this scenario. We need to determine whether the honeybee behavior is actually a good inspiration for our scenario and check whether the approach is efficient. We have at least one crucial parameter, the waiting time depending on temperature, that we need to configure; a model helps to do that in a structured approach. Also for the sake of exercise, of course, it is worth looking into a modeling approach again. But first, we discuss a few alternative sources of inspiration for a modeling approach.

7.4.1 Modeling Aggregation: Interdisciplinary Options

We have not yet developed a generic modeling approach in swarm robotics. Modeling a swarm robotics system is still a kind of an art. Scientifically speaking that is problematic but also an interesting challenge. We should accept the challenge as an opportunity to work interdisciplinary. At the same time, adapting models

from other fields of research also helps to estimate the difficulty of the modeling problem by detecting analogies to other modeling challenges. Aggregation is a perfect example.

Besides investigations of honeybees, there are many more models for aggregation processes in several species: bark beetles [90], ants [91, 379], and cockroaches [7, 91, 163, 199, 234]. Here we go not into further details concerning biological models.

What is even more interesting about modeling aggregation is that a diverse selection of research fields contribute. Especially within physics there are several relevant concepts. Nucleation [342] investigates how a material transforms from one thermodynamic phase to another. Thermodynamic phases are gas, liquid, solid, and plasma. An example of nucleation are water droplets that form in supersaturated vapor (e.g., water droplets on your beer bottle in summer). Although this may seem unrelated, nucleation is used as a model to describe aggregation scenarios, such as traffic jams [23].

Another example from thermodynamics is Ostwald ripening [301, 405] that describes the process following after the nucleation. The system does not go immediately into an equilibrium state but instead several clusters (e.g., of water) form. Ostwald ripening basically describes the competition between the different clusters that grow and dissolve. Vinković and Kirman [403] report an interesting study that combines Ostwald ripening with the Schelling model (a model of segregation in human populations). Similar methods can also be used to model city formation and growth [250].

Obviously physics intensively investigates systems subject to gravitation and also clustering can be an effect of gravitation [181, 279]. Examples are star formation [221] and galaxy formation [362]. Particles or—more general—matter subject to gravitation form deterministic systems that are different from swarms in an important feature. Gravitational systems are fully determined by the initial state and cannot break symmetries in contrast to groups of autonomous agents. Imagine a homogeneous distribution of stationary small particles in 2-d space. Now we add two big stationary masses separated by a certain distance. The small particles are attracted by the two masses. However, we can immediately draw a virtual line in the middle between the two big masses. Particles in the left half approach the left mass and particles in the right half approach the right mass. If we ignore any complex dynamics as observed in three-body problems [30], then we can conclude that the small particles will equally separate and aggregate at either one of the two masses. Swarms of robots are different from that, we can program them to deliberately break such symmetries and aggregate at just one spot [168]. In pattern formation processes, in turn, gravity can also serve itself as means to break symmetries, for example, in the case of pattern formation in cells [373].

In summary, one can say that there are plenty of options to get inspiration for swarm robot aggregation models. From a practical engineering point of view, we can be happy with whatever works. Any model that successfully supports us in designing the swarm robot system has its right. Whether there is one generic model approach that always works is an unanswered question of swarm robotics.

7.4.2 Spatial Model

In the following we want to re-use the micro-macro approach of Sect. 5.3.2 and create a microscopic model for an individual robot together with a macroscopic model for the whole swarm based on the Langevin equation and the Fokker–Planck equation. We define a few modeling objectives. The model should

- represent the difference between the ineffective single behavior with "flat gradient information" and the effective behavior of a swarm;
- represent parameters of the agent algorithm directly in the model;
- represent spatial features;
- show quantitative agreement with robot swarm experiments.

Only if the model can distinguish between the ineffective single behavior and the effective swarm behavior, it can display the important swarm effect due to social interaction. A modeling approach in swarm robotics is only useful if the model can, despite its abstraction level, be related to implementation aspects of the control algorithms. Here we investigate an adaptive aggregation scenario and therefore spatial features are of importance. Often models are sufficient that only show a qualitative agreement (e.g., model reflects only general features of reality) with robot experiments, but if we can push for quantitative agreement (e.g., model predictions match directly with measurements in experiment) with reasonable overhead, then we should do so.

7.4.2.1 Microscopic Model

We construct our microscopic model, the Langevin equation, step by step to model the behavior of an individual robot. We start from the general Langevin equation, Eq. (5.23), and replace $\mathbf{A}(\mathbf{R}(t))$ already with a gradient $\nabla P(\mathbf{R}(t))$, yielding

$$\dot{\mathbf{R}}(t) = \alpha \nabla P(\mathbf{R}(t)) + B\mathbf{F}(t). \tag{7.1}$$

We have to define the parameters. Remember that $\mathbf{R}$ is the robot's position, here in 2-d. We use the potential field P to model the temperature field. We define a gradient ascent depending on parameter α. $\nabla P(\mathbf{R}(t))$ gives the gradient of the temperature field at the robot's current position. With $\alpha \in [0, 1]$ we can regulate the intensity of the robot's goal-oriented behavior.

As before, $\mathbf{F}$ represents the stochastic process that models the robot's random behavior. We assume that the robot goes intentionally or unintentionally in random directions. That may seem strange as it does not directly correspond to step 1 of the behavioral model: "move straight forward." However, it turns out to be a viable modeling assumption and simplification. An alternative modeling approach could be the Lévy flight [248], which allows bigger jumps that could represent straight motion. Here, we stick to the Brownian-motion approach, which is also called

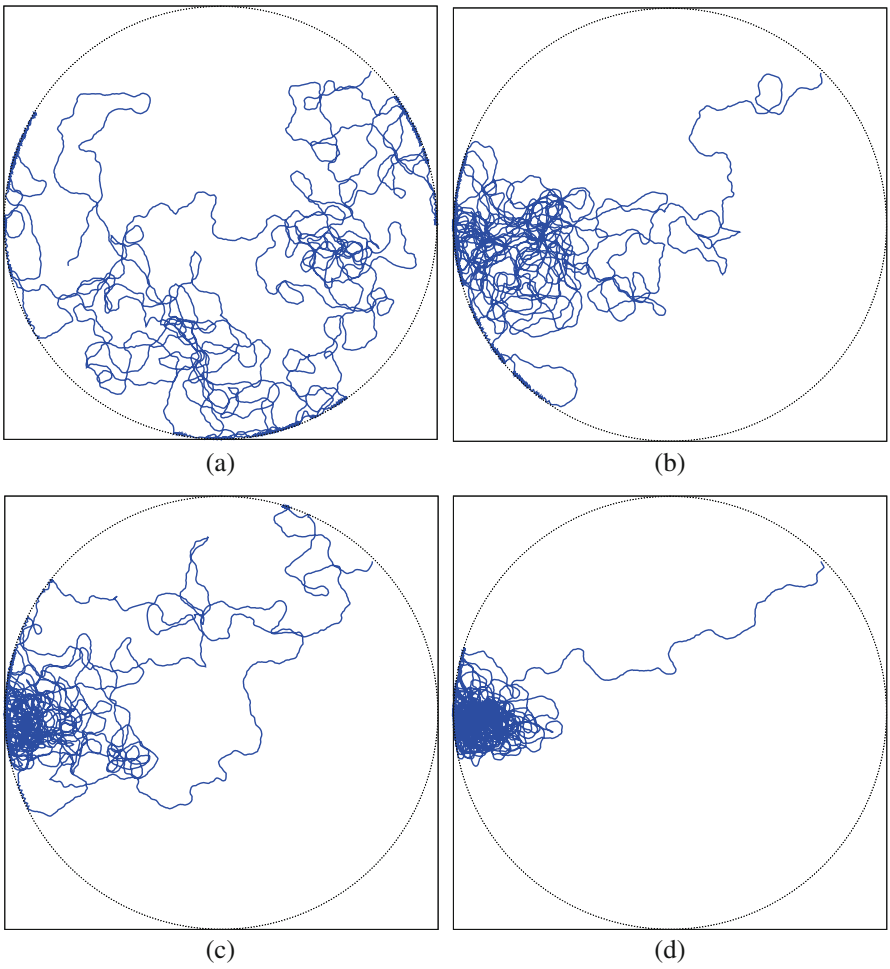


Fig. 7.3 Example trajectories sampled from the Langevin equation, Eq. (7.1), with four settings of parameter $\alpha \in \{0.01, 0.025, 0.05, 0.1\}$ regulating the intensity of the robot's goal-oriented behavior. The desired warm spot is at the left border. (**a**) $\alpha = 0.01$ (**b**) $\alpha = 0.025$ (**c**) $\alpha = 0.05$ (**d**) $\alpha = 0.1$

Wiener process, which in turn is a special case of the Lévy flight. As before, B scales the intensity of the random behavior.

In Fig. 7.3 you see example trajectories using Eq. (7.1) and the effect of varying parameter α. The temperature distribution is still the same as in Fig. 7.1, that is, the desired warm spot is at the left border. For $\alpha = 0.01$ we see basically no bias towards the optimum. Then with increasing α the robot approaches the optimum more and more directly and stays closer to it. Note that we have not yet defined a condition for stopping, hence, the robot keeps moving at all time.

Next, we want to model the "flat gradient information" situation. The question is what the correct setting for parameter α is. In the case of the young honeybee experiments, the situation is somewhat unclear [209, 372]. There are honeybees that approach the warm spot more or less directly (so-called "goal finders") but they vary in their intensity of goal-directedness. At the same time, there exist other young bees that walk around randomly or walk along the boundaries [209, 372]. In order to simplify our approach we are not going to model these diverse results of the biological system and focus instead on possible robot implementations. We go for the rather extreme assumption of ignoring the temperature gradient and the goal-oriented behavior completely. We set $\alpha = 0$. This is a strong assumption because in order to achieve the desired aggregation at the warm spot, we then fully rely on an emergent, macroscopic effect. If we can still manage to aggregate the swarm without individual goal-oriented behavior, then we have also simplified the (hardware) requirements for the robot swarm because robots do not need to detect differences in the temperatures anymore. We set the individual goal-oriented behavior off with $\alpha = 0$ and we get

$$\dot{\mathbf{R}}(t) = \alpha \nabla P(\mathbf{R}(t)) + B\mathbf{F}(t) = B\mathbf{F}(t), \tag{7.2}$$

which means we have a pure random walk. From a modeling perspective, we should appreciate that because it simplified our model considerably.

Now we have to step out of the modeling framework that a mere Langevin equation offers us, because we want to model the stopping behavior of robots. Robots should be allowed to make a state transition from moving to staying stopped. Hence, we identify two states, a robot is either *moving* or *stopped*. For the state *stopped* we need to define the waiting times finally. We define a function, which gives the corresponding waiting time for a given position $\mathbf{R}$ and its associated temperature $P(\mathbf{R})$. Following the standard approach of BEECLUST [344], we define a waiting time function

$$w(\mathbf{R}) = \frac{w_{\max} P^2(\mathbf{R})}{P^2(\mathbf{R}) + c}, \tag{7.3}$$

for maximal waiting time $w_{\max}$, temperature $P(\mathbf{R})$ at position $\mathbf{R}$, and a scaling constant c. See Fig. 7.4 for an example of a waiting time function inspired by the setup based on a light distribution instead of temperature as described by Schmickl and Hamann [344].

We use the waiting time function $w(\mathbf{R})$ to determine for how long a robot has to stay in *stopped* state before it switches back to *moving*. We can model our approach with a finite state machine, see Fig. 7.5. The transition *moving*→*stopped* is triggered by another robot in proximity. We model other robots by additional instances of the microscopic Langevin model. In the moment of that transition, the robot measures the temperature (or light) and determines its waiting time. The waiting time of our model operates via the robot position $\mathbf{R}$, while the robot itself does not know its

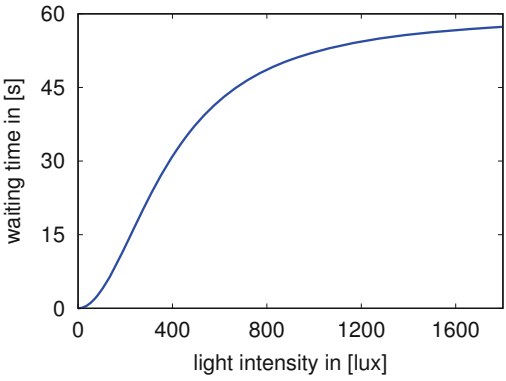


Fig. 7.4 Example waiting time function, Eq. (7.3), for a setting working with light measurements instead of temperature [344]. Parameter setting is $w_{\max} = 60$ and $c = 1.5 \times 10^5$

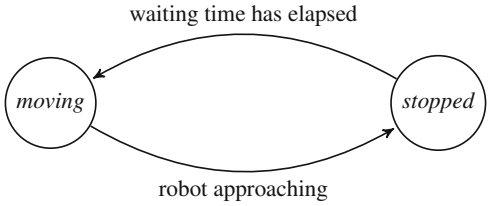


Fig. 7.5 Finite state machine modeling our microscopic modeling approach

position it can certainly measure the local temperature. The backward transition *stopped→moving* is triggered by an elapsed waiting time.

Each of the states is associated with an appropriate Langevin equation that models the robot's dynamics in that state. As described above, for state *moving* we have

$$\dot{\mathbf{R}}_m(t) = B\mathbf{F}(t). \tag{7.4}$$

For state *stopped* it is even simpler, as we have

$$\dot{\mathbf{R}}_s(t) = 0. \tag{7.5}$$

This trivial equation does what it is supposed to: the robot should not move while *stopped*.

Our microscopic model is complete now. The dynamics of the robots is modeled by simple Langevin equations, that we have actually simplified to mere stochastic processes (random walk). The model is not purely mathematical because we have to administrate the state transitions. Also we have a model instance for each robot, so we have to administrate position, state, and remaining waiting time for each robot. In addition, we have a computational overhead because we need to check distances between robots, which generally scales $\mathcal{O}(N^2)$ for swarm size N (unless a domain-specific, special data structure is created). All of that is, however, typical for a microscopic model that is by nature always close to a robot simulation.

7.4.2.2 Macroscopic Model

We construct the macroscopic counterparts to the microscopic model. Thanks to the micro-macro link between the Langevin equation and the Fokker–Planck equation we can do that in a schematic approach. For each microscopic state we define a corresponding probability density. We define $\rho_m(\mathbf{r}, t)$ as the density of moving robots within an infinitesimally small area located at $\mathbf{r}$ at time t. Equivalently, we define $\rho_s(\mathbf{r}, t)$ as density of stopped robots at $\mathbf{r}$ at time t. For both densities we need a model of motion that is directly determined by the corresponding Langevin equation. For state *moving* we get

$$\dot{\mathbf{R}}_m(t) = B\mathbf{F}(t) \quad \Rightarrow \quad \frac{\partial \rho_m(\mathbf{r}, t)}{\partial t} = B^2 \nabla^2 \rho_m(\mathbf{r}, t). \tag{7.6}$$

That is a mere diffusion equation because we deleted any goal-oriented behavior already in the Langevin equation. For state *stopped* we get the expected simple result of

$$\dot{\mathbf{R}}_s(t) = 0 \quad \Rightarrow \quad \frac{\partial \rho_s(\mathbf{r}, t)}{\partial t} = 0. \tag{7.7}$$

In similarity to the finite state machine shown in Fig. 7.5, we need a counterpart for the macroscopic model. The Fokker-Planck approach is continuous and forces us to redefine the transitions. We can do that by extending our Fokker–Planck equation to a so-called master equation (see, for example, the book of Schweitzer [354]). A master equation models the flow of probability density between different states almost as if it would model flows of liquids. So we have to think about how much density flows from the *moving* density to the *stopped* density and back. As said, the microscopic and inherently discrete transitions "waiting time elapses" and "robot approaching" need to be translated into continuous counterparts.

In the case of the transition *moving*→*stopped* we cannot determine individual robots approaching each other because we deal only with probability densities. We have to determine a probabilistic approach basically asking: given density $\rho_m(\mathbf{r}, t)$, how likely is it that (at least) two robots approach each other. In order to keep this modeling approach here simple, we make a relatively strong assumption saying that the probability of two robots approaching each other is directly proportional to the density of moving robots $\rho_m(\mathbf{r}, t)$. Hence, we can simply define a stopping rate φ. We say that the product $\rho_m(\mathbf{r}, t)\varphi$ gives us the correct density flow of stopping robots—a rather crude approach hiding all details of how an effective model can be created this way [165, 346].

It turns out, we can model the flow for *stopped*→*moving* with a so-called delay equation (similar to what we did with rate equations in Sect. 5.3.1). So with the waiting time function we can determine a time delay $w(\mathbf{r})$. That means that robots $\rho_m(\mathbf{r}, t)\varphi$ leaving state *moving* spend $w(\mathbf{r})$ time in a waiting loop until

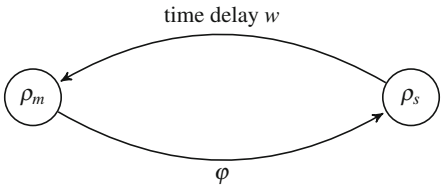


Fig. 7.6 Finite state machine modeling our macroscopic modeling approach

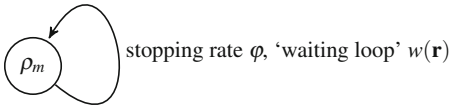


Fig. 7.7 Simplified finite state machine modeling our macroscopic approach with a single robot density $\rho_m(\mathbf{r}, t)$ and time delays

they arrive back in state *moving* again. The resulting finite state machine for the macroscopic model is shown in Fig. 7.6.

Actually the density of stopped robots ρ_s is mathematically unnecessary. Robots leaving state *moving* just spend some time in "nirvana" and come back again. One can also say, they stay in a waiting loop for time $w(\mathbf{r})$ (almost like an airplane waiting for clearance to land). So we delete ρ_s and use time delays instead as shown in Fig. 7.7. The swarm fraction that is subtracted from $\rho_m(\mathbf{r}, t_0)$ at time t_0 is added to $\rho_m(\mathbf{r}, t_1)$ again at time $t_1 = t_0 + w(\mathbf{r})$.

The resulting macroscopic model is completely described by the time evolution of the density of robots in state *moving*, given by

$$\frac{\partial \rho_m(\mathbf{r}, t)}{\partial t} = B^2 \nabla^2 \rho_m(\mathbf{r}, t) - \rho_m(\mathbf{r}, t)\varphi + \rho_m(\mathbf{r}, t - w(\mathbf{r}))\varphi, \tag{7.8}$$

with stopping rate φ as described above and we would need to determine B experimentally.

As said, it is not necessary but for completeness and a better understanding we also give the density of robots in state *stopped*

$$\frac{\partial \rho_s(\mathbf{r}, t)}{\partial t} = \rho_m(\mathbf{r}, t)\varphi - \rho_m(\mathbf{r}, t - w(\mathbf{r}))\varphi. \tag{7.9}$$

With Eq. (7.8) we have now a model. In a usual use case we apply it in the following way. We input a relevant initial state, which is here a spatial distribution of robots in state *moving*. Due to the time delay $t - w(\mathbf{r})$ we also have to define values for the recent past before "time zero" of our model to reach a complete description of an initial state. Typically, one would set that history to $\rho_m(\mathbf{r}, t) = 0$ for $t < 0$ to keep things simple. That is, we assume nobody ever stopped in this system before. Mathematically we define an initial value problem that we need to solve now. Many

PDE and especially PDE with time-delays are not easily solved analytically. We restrict ourselves to a numerical solution that we do not want to describe in detail here. A simple approach is a forward integration in time with the Euler method. For more details see, for example, Press et al. [316] and Hamann [165].

7.5 Verification

Finally, we want to verify our model, Eq. (7.8), and our algorithm (see Sect. 7.3) with robot experiments. We follow closely the study by Schmickl et al. [346] and the similar study by Kernbach et al. [212].

In the robot experiments the Jasmine [197] robot was used. Jasmine is a simple and small swarm robot ($30\times30\times25\,\mathrm{mm}^3$) with differential drive. It has six infrared sensors pointing all around the robot. On top they have two sensors measuring the ambient light. The swarm has a size of 15 Jasmine robots that operate on a rectangular area of $150\times100\,\mathrm{cm}^2$. The spatial temperature distribution of the young honeybees is replaced by a light distribution. There is one lamp at each of the two short sides of the arena. The lamps are operated in three modes each: off, dimmed, and bright. The robot experiment is structured in four phases to test the robot swarm's reaction to a dynamic environment. This approach is also helpful to verify the models. In each of the four phases the two lights—left, right—are set to

- off, dimmed;
- bright, dimmed;
- dimmed, bright;
- dimmed, off.

Each phase lasts 3 min (180 s). The robot controller is an implementation of the behavioral model for the young honeybees described in Sect. 7.3. A notable detail is that the robots do not reliably detect each other. Both detection of obstacles (walls) and detection of robots are based on infrared. So robots have to discriminate between the two. This is done by sophisticated timing. If the robot currently measures distances with its infrared sensors, then incoming infrared signals are interpreted as reflections. If the robot is currently not measuring distances but receiving infrared signals, then it assumes a closeby robot.

Six independent experiments with 15 Jasmine robots over the full duration of 720 s were done. Initially, the robots were distributed uniformly randomly all over the rectangular area and were set to state *moving*. Similarly, the macroscopic Fokker–Planck model, Eq. (7.8), was initialized to a uniform distribution with all density in state *moving*. The efficiency of the aggregation behavior in the robot experiments was measured by counting stopped robots beneath the two lights, for details see Schmickl et al. [346]. In the case of the Fokker–Planck model, the initial value problem was solved by forward integration in time. To measure the equivalent of aggregated robots at any time, the density of the respective area beneath the lamps was integrated; for details see Schmickl and Hamann [344]. The

model was fitted using the single free parameter B in Eq. (7.8), that is, the intensity of random behavior. Note that this intensity of random behavior is influenced by the environment, the robot behavior, and the robot–robot encounters. The more often robots approach each other the more often they turn, which influences their distribution in space. That distribution is modeled here exclusively by a diffusion process. Hence, we need to find the parameter B experimentally, which is done here post-hoc by merely fitting the model to the robot experiments. Although fitting a free parameter of course helps to get a better agreement between model and experiment, the results given in Fig. 7.8 show still an unexpected good match between the robot experiments and the Fokker–Planck approach.

Initially robots aggregate at the dimmed right light. After 180 s the left light is turned on to bright mode, gathers quickly aggregated robots, and the cluster at the

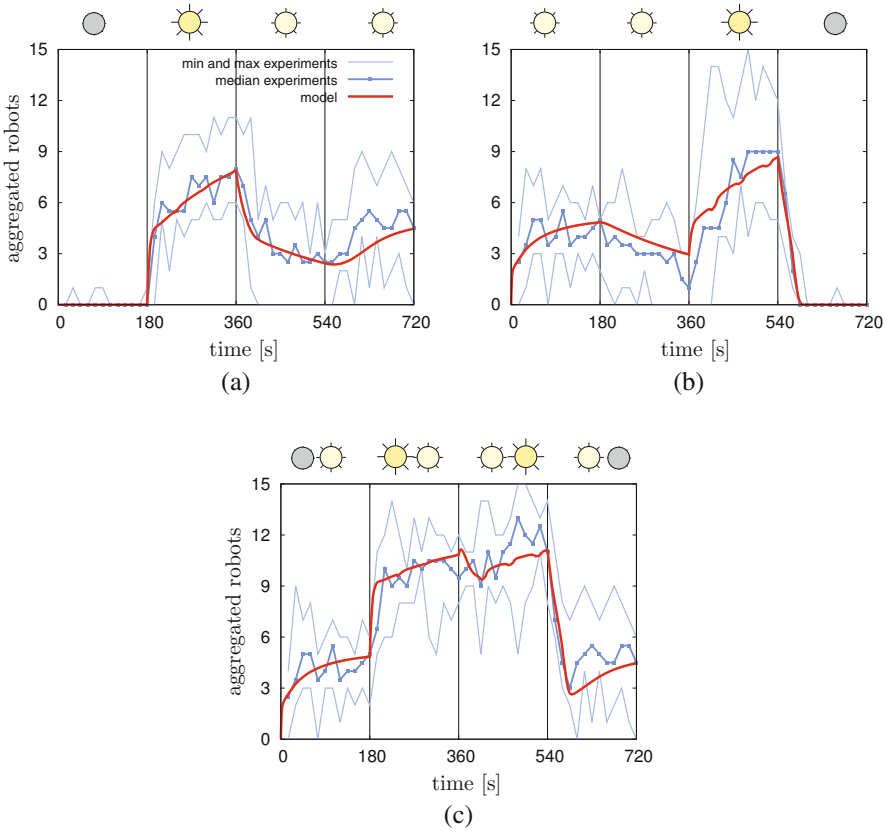


Fig. 7.8 Number of aggregated robots beneath the left, the right, and any lamp. The blue lines gives results (min., median, and max.) of six independent robot experiments with 15 Jasmine robots. The red line gives the result of a Fokker–Planck model given in Eq. (7.8) [165, 346]. (**a**) Aggregated robots at the left light (**b**) Aggregated robots at the right light (**c**) Sum of aggregated robots at both lights

right light is reduced. Then the right light is increased to bright mode and the left light is reduced to dimmed mode. As expected, the right light gathers additional robots again and the left light loses robots. Finally, starting from $t = 540$, more robots gather at the left light and the cluster at the right light dissolves.

The model represents all dynamics qualitatively correctly and the quantitative agreement is also good. Note that the free parameter B basically influences the speed of the diffusion process but has limited influence on these qualitative aspects. For more details about the spatial predictions of the model see, for example, Correll and Hamann [74] and Schmickl et al. [346].

In an actual application, we can use the model, for example, to refine the waiting time function $w(\mathbf{R})$. We may have certain requirements, such as a quick cluster formation or a clear distinction between cluster sizes at dimmed lights versus bright lights. We can use the model to test quickly many different setups, quantify their performance automatically, and pick appropriate parameters. Also different temperature or light fields can be tested. The results gained by the model, however, need to be verified in simulations and eventually in hardware experiments. The model may be of high quality but it is still an abstraction that can by principle not represent all real situations.

7.6 Short Summary

In this chapter we tried to go through the whole swarm robot design process in one effort. We started from the specification by our customer and some bio-inspiration luckily helped us in defining an effective algorithm. We have found a control algorithm that makes the swarm adaptive to a dynamic environment. We have developed a spatial model that can model where robots tend to aggregate depending on environmental features and parameters of the algorithm. The model is in good agreement with robot experiments and we could use it to optimize the waiting time function and the stopping rate. So let's hope our customer is satisfied.

7.7 Further Reading

Where to go from here? Actually I'm convinced that you can make your way alone from here. Feel free to explore the rich swarm robotics literature for yourself. Still, here are some hints about literature directly related to this chapter's content. The literature on BEECLUST is big and versatile. Starting from Schmickl and Hamann [344] you can read the more recent works by Arvin et al. [17, 18] and Wahby [406]. There are also interesting studies of several interacting swarms by Meister [269] and Bodi [44, 45]. There are many other interesting works on aggregation in robots and animals: Trianni et al. [384], Soysal and Şahin [361], and Garnier et al. [137].

Epilogue

> *Skepticism is like a microscope whose magnification is constantly increased: the sharp image that one begins with finally dissolves, because it is not possible to see ultimate things: their existence is only to be inferred.*
>
> —Stanisław Lem, His Master's Voice

There is not much left to say. Hopefully, this book has triggered your curiosity and you feel urged to test your ideas. Swarm robotics is still a young field and there is still so much to be revealed. With the uprise of robotics within the next decades we will see more and more implementations and applications of swarm robotics in the field and in real-world applications. It is an exciting time for swarm robotics research. From the theoretical perspective there seem to be many hard problems left that wait for the right moment and person to solve them. Maybe there exists even a unifying theory that solves the micro-macro problem and that explains how we should design swarm robotics systems. To conclude I quote an enthusiastic text of Ed Yong[1]:

> All these similarities seem to point to a grand unified theory of the swarm—a fundamental ultra-calculus that unites the various strands of group behavior. In one paper, Vicsek and a colleague wondered whether there might be "some simple underlying laws of nature (such as, e.g., the principles of thermodynamics) that produce the whole variety of the observed phenomena."

Please correct my errors and spread your knowledge.

[1]Ed Yong, *How the Science of Swarms Can Help Us Fight Cancer and Predict the Future* on Wired.com, https://www.wired.com/2013/03/powers-of-swarms/, 2013.

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2

References

1. Abbott, R. (2004). Emergence, entities, entropy, and binding forces. In *The Agent 2004 Conference on: Social Dynamics: Interaction, Reflexivity, and Emergence*, Argonne National Labs and University of Chicago, October 2004.
2. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, T., et al. (2000). Amorphous computing. *Communications of the ACM, 43*(5), 74–82.
3. Adamatzky, A. (2010). *Physarum machines: Computers from slime mould*. Singapore: World Scientific.
4. Akhtar, N., Ozkasap, O., & Ergen, S. C. (2013). Vanet topology characteristics under realistic mobility and channel models. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1774–1779). New York: IEEE.
5. Allwright, M., Bhalla, N., & Dorigo, M. (2017). Structure and markings as stimuli for autonomous construction. In *2017 18th International Conference on Advanced Robotics (ICAR)* (pp. 296–302). New York: IEEE.
6. Allwright, M., Bhalla, N., El-faham, H., Antoun, A., Pinciroli, C., & Dorigo, M. (2014). SRoCS: Leveraging stigmergy on a multi-robot construction platform for unknown environments. In *International Conference on Swarm Intelligence* (pp. 158–169). Cham: Springer.
7. Ame, J.-M., Rivault, C., & Deneubourg, J.-L. (2004). Cockroach aggregation based on strain odour recognition. *Animal Behaviour, 68*, 793–801.
8. Anderson, C., Boomsma, J. J., & Bartholdi, J. J. (2002). Task partitioning in insect societies: Bucket brigades. *Insectes Sociaux, 49*, 171–180.
9. Anderson, C., Theraulaz, G., & Deneubourg, J.-L. (2002). Self-assemblages in insect societies. *Insectes Sociaux, 49*(2), 99–110.
10. Anderson, P. W. (1972). More is different. *Science, 177*(4047), 393–396.
11. Arbuckle, D. J. (2007). *Self-assembly and Self-repair by Robot Swarms*, University of Southern California.
12. Arbuckle, D. J., & Requicha, A. A. G. (2010). Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: Algorithms and simulations. *Autonomous Robots, 28*(2), 197–211. ISSN 1573-7527. https://doi.org/10.1007/s10514-009-9162-7
13. Arkin, R. C. (1998). *Behavior-based robotics*. Cambridge, MA: MIT Press.
14. Arkin, R. C., & Egerstedt, M. (2015). Temporal heterogeneity and the value of slowness in robotic systems. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (pp. 1000–1005). New York: IEEE.
15. Arthur, W. B. (1989). Competing technologies, increasing returns, and lock-in by historical events. *The Economic Journal, 99*(394), 116–131.

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2

16. Arvin, F., Murray, J., Zhang, C., & Yue, S. (2014). Colias: An autonomous micro robot for swarm robotic applications. *International Journal of Advanced Robotic Systems, 11*(7), 113. http://dx.doi.org/10.5772/58730
17. Arvin, F., Turgut, A. E., Bazyari, F., Arikan, K. B., Bellotto, N., & Yue, S. (2014). Cue-based aggregation with a mobile robot swarm: A novel fuzzy-based method. *Adaptive Behavior, 22*(3), 189–206.
18. Arvin, F., Turgut, A. E., Krajník, T., & Yue, S. (2016). Investigation of cue-based aggregation in static and dynamic environments with a mobile robot swarm. *Adaptive Behavior, 24*(2), 102–118. https://doi.org/10.1177/1059712316632851
19. Arvin, F., Turgut, A. E., & Yue, S. (2012). Fuzzy-based aggregation with a mobile robot swarm. In *Swarm intelligence (ANTS'12). Lecture notes in computer science* (Vol. 7461, pp. 346–347). Berlin: Springer. ISBN 978-3-642-32649-3. https://doi.org/10.1007/978-3-642-32650-9_39
20. Augugliaro, F., Lupashin, S., Hamer, M., Male, C., Hehn, M., Mueller, M. W., et al. (2014). The flight assembled architecture installation: Cooperative construction with flying machines. *IEEE Control Systems, 34*(4), 46–64.
21. Axelrod, R. (1997). The dissemination of culture: A model with local convergence and global polarization. *Journal of Conflict Resolution, 41*(2), 203–226.
22. Bachrach, J., & Beal, J. (2006). Programming a sensor network as an amorphous medium. In *Distributed computing in sensor systems (DCOSS'06, extended abstract)*.
23. Bak, P. (2013). *How nature works: The science of self-organized criticality*. Berlin: Springer Science & Business Media.
24. Bak, P., Tang, C., & Wiesenfeld, K. (1988). Self-organized criticality. *Physical Review A, 38*(1), 364–374. https://doi.org/10.1103/PhysRevA.38.364
25. Balch, T. (2000). Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous Robots, 8*(3), 209–238. ISSN 0929-5593.
26. Ball, P. (2015). Forging patterns and making waves from biology to geology: A commentary on Turing (1952) 'the chemical basis of morphogenesis'. *Philosophical Transactions of the Royal Society of London B: Biological Sciences, 370*(1666), 20140218. ISSN 0962-8436. https://doi.org/10.1098/rstb.2014.0218
27. Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., et al. (2008). Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *Proceedings of the National Academy of Sciences of the United States of America, 105*(4), 1232–1237.
28. Baluška, F., & Levin, M. (2016). On having no head: Cognition throughout biological systems. *Frontiers in Psychology, 7*, 902. ISSN 1664-1078. https://www.frontiersin.org/article/10.3389/fpsyg.2016.00902
29. Baran, P. (1960). *Reliable digital communications systems using unreliable network repeater nodes*. Technical report, The RAND Corporation, Santa Monica, CA.
30. Barrow-Green, J. (1997). *Poincaré and the three body problem*. American Mathematical Society, London: London Mathematical Society.
31. Bass, F. M. (1969). A new product growth for model consumer durables. *Management Science, 15*(5), 215–227.
32. Bayindir, L. (2015). A review of swarm robotics tasks. *Neurocomputing, 172*(C), 292–321. http://dx.doi.org/10.1016/j.neucom.2015.05.116
33. Bayindir, L., & Şahin, E. (2007). A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering and Computer Sciences, 15*, 115–147. http://journals.tubitak.gov.tr/elektrik/issues/elk-07-15-2/elk-15-2-2-0705-13.pdf
34. Beckers, R., Holland, O. E., & Deneubourg, J.-L. (1994). From local actions to global tasks: Stigmergy and collective robotics. In *Artificial life IV* (pp. 189–197). Cambridge, MA: MIT Press.
35. Bekey, G., Ambrose, R., Kumar, V., Lavery, D., Sanderson, A., Wilcox, B., et al. (2008). *Robotics: State of the art and future challenges*. Singapore: World Scientific.

36. Belykh, I., Jeter, R., & Belykh, V. (2017). Foot force models of crowd dynamics on a wobbly bridge. *Science Advances, 3*(11), e1701512. https://doi.org/10.1126/sciadv.1701512
37. Beni, G. (2005). From swarm intelligence to swarm robotics. In E. Şahin & W. M. Spears (Eds.), *Swarm Robotics - SAB 2004 International Workshop*, Santa Monica, CA, July 2005. *Lecture notes in computer science* (Vol. 3342, pp. 1–9). Berlin: Springer. http://dx.doi.org/10.1007/978-3-540-30552-1_1
38. Berea, A., Cohen, I., D'Orsogna, M. R., Ghosh, K., Goldenfeld, N., Goodnight, C. J., et al. (2014). IDR team summary 6. In *Collective behavior: From cells to societies*. Washington, DC: The National Academies Press.
39. Berg, B. A., & Billoire, A. (2007). Markov chain Monte Carlo simulations. In B. W. Wah (Ed.), *Wiley encyclopedia of computer science and engineering*. New York: Wiley. https://dx.doi.org/10.1002/9780470050118.ecse696
40. Berman, S., Lindsey, Q., Sakar, M. S., Kumar, V., & Pratt, S. C. (2011). Experimental study and modeling of group retrieval in ants as an approach to collective transport in swarm robotic systems. *Proceedings of the IEEE, 99*(9), 1470–1481. ISSN 0018-9219. https://doi.org/10.1109/JPROC.2011.2111450
41. Biancalani, T., Dyson, L., & McKane, A. J. (2014). Noise-induced bistable states and their mean switching time in foraging colonies. *Physical Review Letters, 112*, 038101. http://link.aps.org/doi/10.1103/PhysRevLett.112.038101
42. Bjerknes, J. D., & Winfield, A. (2013). On fault-tolerance and scalability of swarm robotic systems. In A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, M. Ani Hsieh, L. E. Parker, & K. Støy (Eds.), *Distributed autonomous robotic systems (DARS 2010). Springer tracts in advanced robotics* (Vol. 83, pp. 431–444). Berlin: Springer. ISBN 978-3-642-32722-3. http://dx.doi.org/10.1007/978-3-642-32723-0_31
43. Bjerknes, J. D., Winfield, A., & Melhuish, C. (2007). An analysis of emergent taxis in a wireless connected swarm of mobile robots. In Y. Shi & M. Dorigo (Eds.), *IEEE Swarm Intelligence Symposium*, Los Alamitos, CA (pp. 45–52). New York: IEEE Press.
44. Bodi, M., Thenius, R., Schmickl, T., & Crailsheim, K. (2011). How two cooperating robot swarms are affected by two conflictive aggregation spots. In *Advances in Artificial Life: Darwin Meets von Neumann (ECAL'09). Lecture notes in computer science* (Vol. 5778, pp. 367–374). Heidelberg/Berlin: Springer.
45. Bodi, M., Thenius, R., Szopek, M., Schmickl, T., & Crailsheim, K. (2011). Interaction of robot swarms using the honeybee-inspired control algorithm BEECLUST. *Mathematical and Computer Modelling of Dynamical Systems, 18*, 87–101. http://www.tandfonline.com/doi/abs/10.1080/13873954.2011.601420
46. Bogacz, R., Brown, E., Moehlis, J., Holmes, P., & Cohen, J. D. (2006). The physics of optimal decision making: A formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological Review, 113*(4), 700.
47. Bonabeau, E. (2002). Predicting the unpredictable. *Harvard Business Review, 80*(3), 109–116.
48. Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. New York, NY: Oxford University Press.
49. Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., et al. (2010). The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4187–4193). New York: IEEE.
50. Bongard, J. C. (2013). Evolutionary robotics. *Communications of the ACM, 56*(8), 74–83. http://doi.acm.org/10.1145/2493883
51. Bonnet, F., Cazenille, L., Gribovskiy, A., Halloy, J., & Mondada, F. (2017). Multi-robot control and tracking framework for bio-hybrid systems with closed-loop interaction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017 (pp. 4449–4456). https://doi.org/10.1109/ICRA.2017.7989515
52. Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology*. Cambridge, MA: MIT Press.

53. Brambilla, M. (2014). *Formal Methods for the Design and Analysis of Robot Swarms*. PhD thesis, Université Libre de Bruxelles.
54. Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence, 7*(1), 1–41. ISSN 1935-3812. http://dx.doi.org/10.1007/s11721-012-0075-2
55. Breder, C. M. (1954). Equations descriptive of fish schools and other animal aggregations. *Ecology, 35*(3), 361–370.
56. Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation, 2*(1), 14–23.
57. Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence, 47*, 139–159.
58. Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., & Dorigo, M. (2014). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems, 28*(1), 101–125. ISSN 1387-2532. http://dx.doi.org/10.1007/s10458-012-9212-y
59. Buhl, J., Sumpter, D. J. T., Couzin, I. D., Hale, J. J., Despland, E., Miller, E. R., & Simpson, S. J. (2006). From disorder to order in marching locusts. *Science, 312*(5778), 1402–1406. https://doi.org10.1126/science.1125142
60. Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., & Bonabeau, E. (2001). *Self-organizing biological systems*. Princeton, NJ: Princeton University Press.
61. Campo, A., Garnier, S., Dédriche, O., Zekkri, M., & Dorigo, M. (2011). Self-organized discrimination of resources. *PLoS One, 6*(5), e19888.
62. Caprari, G., Balmer, P., Piguet, R., & Siegwart, R. (1998). The autonomous microbot 'Alice': A platform for scientific and commercial applications. In *Proceedings of the Ninth International Symposium on Micromechatronics and Human Science*, Nagoya, Japan (pp. 231–235).
63. Caprari, G., Colot, A., Siegwart, R., Halloy, J., & Deneubourg, J.-L. (2005). Animal and robot mixed societies: Building cooperation between microrobots and cockroaches. *IEEE Robotics & Automation Magazine, 12*(2), 58–65. https://doi.org/10.1109/MRA.2005.1458325
64. Carlsson, H., & Van Damme, E. (1993). Global games and equilibrium selection. *Econometrica: Journal of the Econometric Society, 6*(5), 989–1018.
65. Carneiro, J., Leon, K., Caramalho, Í., Van Den Dool, C., Gardner, R., Oliveira, V., et al. (2007). When three is not a crowd: A crossregulation model of the dynamics and repertoire selection of regulatory cd4+ t cells. *Immunological Reviews, 216*(1), 48–68.
66. Castellano, C., Fortunato, S., & Loreto, V. (2009). Statistical physics of social dynamics. *Reviews of Modern Physics, 81*, 591–646. https://link.aps.org/doi/10.1103/RevModPhys.81.591
67. Cavalcanti, A., & Freitas, R. A. Jr. (2005). Nanorobotics control design: A collective behavior approach for medicine. *IEEE Transactions on NanoBioscience, 4*(2), 133–140.
68. Chaimowicz, L., & Kumar, V. (2007). Aerial shepherds: Coordination among UAVs and swarms of robots. In *Distributed Autonomous Robotic Systems 6* (pp. 243–252). Berlin: Springer.
69. Chazelle, B. (2015). An algorithmic approach to collective behavior. *Journal of Statistical Physics, 158*(3), 514–548.
70. Chen, J., Gauci, M., Price, M. J., & Groß, R. (2012). Segregation in swarms of e-puck robots based on the Brazil nut effect. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Richland, SC (pp. 163–170). IFAAMAS.
71. Christensen, A. L., O'Grady, R., & Dorigo, M. (2009). From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation, 13*(4), 754–766. ISSN 1089-778X. https://doi.org/10.1109/TEVC.2009.2017516
72. Ciocchetta, F., & Hillston, J. (2009). Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science, 410*(33), 3065–3084.
73. Clifford, P., & Sudbury, A. (1973). A model for spatial conflict. *Biometrika, 60*(3), 581–588. https://doi.org/10.1093/biomet/60.3.581.

74. Correll, N., & Hamann, H. (2015). Probabilistic modeling of swarming systems. In J. Kacprzyk & W. Pedrycz (Eds.) *Springer handbook of computational intelligence* (pp. 1423–1431). Berlin: Springer.
75. Correll, N., Schwager, M., & Rus, D. (2008). Social control of herd animals by integration of artificially controlled congeners. In *Proceedings of the 10th International Conference on Simulation of Adaptive Behavior: From Animals to Animats*. *Lecture notes in computer science* (Vol. 5040, pp. 437–446). Berlin: Springer.
76. Couzin, I. D., Ioannou, C. C., Demirel, G., Gross, T., Torney, C. J., Hartnett, A., et al. (2011). Uninformed individuals promote democratic consensus in animal groups. *Science, 334*(6062), 1578–1580. ISSN 0036-8075. https://doi.org/10.1126/science.1210280. http://science.sciencemag.org/content/334/6062/1578
77. Couzin, I. D., Krause, J., Franks, N. R., & Levin, S. A. (2005). Effective leadership and decision-making in animal groups on the move. *Nature, 433*, 513–516.
78. Couzin, I. D., Krause, J., James, R., Ruxton, G. D., & Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology, 218*, 1–11. https://doi.org/10.1006/jtbi.2002.3065
79. Crespi, V., Galstyan, A., & Lerman, K. (2008). Top-down vs bottom-up methodologies in multi-agent system design. *Autonomous Robots, 24*(3), 303–313.
80. Crutchfield, J. (1994). The calculi of emergence: Computation, dynamics, and induction. *Physica D, 75*(1–3), 11–54.
81. Crutchfield, J. (1994). Is anything ever new? In G. Cowan, D. Pines, & D. Melzner (Eds.), *Complexity: metaphors, models, and reality*. *SFI series in the sciences of complexity proceedings* (Vol. 19, pp. 479–497). Reading, MA: Addison-Wesley.
82. da Silva Guerra, R., Aonuma, H., Hosoda, K., & Asada, M. (2010). Behavior change of crickets in a robot-mixed society. *Journal of Robotics and Mechatronics, 22*, 526–531.
83. Darley, V. (1994). Emergent phenomena and complexity. In R. Brooks & P. Maes (Eds.), *Artificial life IV* (pp. 411–416). Cambridge, MA: MIT Press.
84. De Nardi, R., & Holland, O. E. (2007). Ultraswarm: A further step towards a flock of miniature helicopters. In E. Şahin, W. M. Spears, & A. F. T. Winfield (Eds.), *Swarm Robotics - Second SAB 2006 International Workshop*. *Lecture notes in computer science* (Vol. 4433, pp. 116–128). Berlin: Springer.
85. De Nicola, R., Ferrari, G. L., & Pugliese, R. (1998). KLAIM: A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering, 24*(5), 315–330.
86. De Nicola, R., Katoen, J., Latella, D., Loreti, M., & Massink, M. (2007). Model checking mobile stochastic logic. *Theoretical Computer Science, 382*(1), 42–70.
87. de Oca, M. M., Ferrante, E., Scheidler, A., Pinciroli, C., Birattari, M., & Dorigo, M. (2011). Majority-rule opinion dynamics with differential latency: A mechanism for self-organized collective decision-making. *Swarm Intelligence, 5*, 305–327. ISSN 1935-3812. http://dx.doi.org/10.1007/s11721-011-0062-z
88. De Wolf, T., & Holvoet, T. (2005). Emergence versus self-organisation: Different concepts but promising when combined. In S. Brueckner, G. D. M. Serugendo, A. Karageorgos, & R. Nagpal (Eds.), *Proceedings of the Workshop on Engineerings Self Organising Applications*. *Lecture notes in computer science* (Vol. 3464, pp. 1–15). Berlin: Springer.
89. Deguet, J., Demazeau, Y., & Magnin, L. (2006). Elements about the emergence issue: A survey of emergence definitions. *Complexus, 3*(1–3), 24–31.
90. Deneubourg, J.-L., Gregoire, J.-C., & Le Fort, E. (1990). Kinetics of larval gregarious behavior in the bark beetle *Dendroctonus micans* (coleoptera: Scolytidae). *Journal of Insect Behavior, 3*(2), 169–182.
91. Deneubourg, J.-L., Lioni, A., & Detrain, C. (2002). Dynamics of aggregation and emergence of cooperation. *Biological Bulletin, 202*, 262–267.
92. Deng, B. (2015). Machine ethics: The robot's dilemma. *Nature, 523*, 24–66. http://dx.doi.org/10.1038/523024a

93. Ding, H., & Hamann, H. (2014). Sorting in swarm robots using communication-based cluster size estimation. In M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. M. de Oca, C. Solnon, & T. Stützle (Eds.), *Ninth International Conference on Swarm Intelligence (ANTS 2014). Lecture notes in computer science* (Vol. 8667, pp. 262–269). Berlin: Springer.
94. Divband Soorati, M., & Hamann, H. (2016). Robot self-assembly as adaptive growth process: Collective selection of seed position and self-organizing tree-structures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)* (pp. 5745–5750). New York: IEEE. http://dx.doi.org/10.1109/IROS.2016.7759845
95. Dixon, C., Winfield, A., & Fisher, M. (2011). Towards temporal verification of emergent behaviours in swarm robotic systems. *Towards Autonomous Robotic Systems (TAROS)* (pp. 336–347). Berlin: Springer.
96. Dorigo, M., Birattari, M., & Brambilla, M. (2014). Swarm robotics. *Scholarpedia, 9*(1), 1463. http://dx.doi.org/10.4249/scholarpedia.1463
97. Dorigo, M., Bonabeau, E., & Theraulaz, G. (2000). Ant algorithms and stigmergy. *Future Generation Computer Systems, 16*(9), 851–871.
98. Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., et al. (2013). Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine, 20*(4), 60–71.
99. Dorigo, M., & Şahin, E. (2004). Guest editorial: Swarm robotics. *Autonomous Robots, 17*(2–3), 111–113.
100. Dorigo, M., Trianni, V., Sahin, E., Groß, R., Labella, T. H., Baldassarre, G., et al. (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots, 17*, 223–245. https://doi.org/10.1023/B:AURO.0000033972.50769.1c.
101. Dorigo, M., Tuci, E., Trianni, V., Groß, R., Nouyan, S., Ampatzis, C., et al. (2006). SWARM-BOT: Design and implementation of colonies of self-assembling robots. In G. Y. Yen & D. B. Fogel (Eds.), *Computational intelligence: Principles and practice* (pp. 103–135). Los Alamitos, CA: IEEE Press.
102. Douven, I., & Riegler, A. (2009). Extending the Hegselmann–Krause model I. *Logic Journal of IGPL, 18*(2), 323–335.
103. Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., et al. (2016). Evolution of collective behaviors for a real swarm of aquatic surface robots. *PLoS One, 11*(3), 1–25. https://doi.org/10.1371/journal.pone.0151834.
104. Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., et al. (2016). Unleashing the potential of evolutionary swarm robotics in the real world. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, New York, NY, USA (pp. 159–160). New York: ACM. ISBN 978-1-4503-4323-7. http://doi.acm.org/10.1145/2908961.2930951
105. Ducatelle, F., Di Caro, G. A., & Gambardella, L. M. (2010). Cooperative self-organization in a heterogeneous swarm robotic system. In *Proceedings of the 12th Conference on Genetic and Evolutionary Computation (GECCO)* (pp. 87–94). New York: ACM.
106. Dussutour, A., Beekman, M., Nicolis, S. C., & Meyer, B. (2009). Noise improves collective decision-making by ants in dynamic environments. *Proceedings of the Royal Society London B, 276*, 4353–4361.
107. Dussutour, A., Fourcassié, V., Helbing, D., & Deneubourg, J.-L. (2004). Optimal traffic organization in ants under crowded conditions. *Nature, 428*, 70–73.
108. Ehrenfest, P., & Ehrenfest, T. (1907). Über zwei bekannte Einwände gegen das Boltzmannsche H-Theorem. *Physikalische Zeitschrift, 8*, 311–314.
109. Eiben, Á. E., & Smith, J. E. (2003). *Introduction to evolutionary computing*. *Natural computing series*. Berlin: Springer.
110. Eigen, M., & Schuster, P. (1977). A principle of natural self-organization. *Naturwissenschaften, 64*(11), 541–565. ISSN 0028-1042. http://dx.doi.org/10.1007/BF00450633

111. Eigen, M., & Schuster, P. (1979). *The hypercycle: A principle of natural self organization*. Berlin: Springer.
112. Eigen, M., & Winkler, R. (1993). *Laws of the game: How the principles of nature govern chance*. Princeton, NJ: Princeton University Press. ISBN 978-0-691-02566-7.
113. Elmenreich, W., Heiden, B., Reiner, G., & Zhevzhyk, S. (2015). A low-cost robot for multi-robot experiments. In *12th International Workshop on Intelligent Solutions in Embedded Systems (WISES)* (pp. 127–132). New York: IEEE.
114. Erdmann, U., Ebeling, W., Schimansky-Geier, L., & Schweitzer, F. (2000). Brownian particles far from equilibrium. *The European Physical Journal B - Condensed Matter and Complex Systems, 15*(1), 105–113.
115. Erdős, P., & Rényi, A. (1959). On random graphs. *Publicationes Mathematicae Debrecen, 6*(290–297), 156.
116. Farrow, N., Klingner, J., Reishus, D., & Correll, N. (2014). Miniature six-channel range and bearing system: Algorithm, analysis and experimental validation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6180–6185). New York: IEEE.
117. Ferber, J. (1999). *Multi-agent systems: An introduction to distributed artificial intelligence*. New York: Addison-Wesley.
118. Ferrante, E., Dúeñez Guzmán, E., Turgut, A. E., & Wenseleers, T. (2013). Evolution of task partitioning in swarm robotics. In V. Trianni (Ed.), *Proceedings of the Workshop on Collective Behaviors and Social Dynamics of the European Conference on Artificial Life (ECAL 2013)*. Cambridge, MA: MIT Press.
119. Ferrante, E., Turgut, A. E., Duéñez-Guzmàn, E., Dorigo, M., & Wenseleers, T. (2015). Evolution of self-organized task specialization in robot swarms. *PLoS Computational Biology, 11*(8), e1004273. https://doi.org/10.1371/journal.pcbi.1004273.
120. Ferrer, E. C. (2016). The blockchain: A new framework for robotic swarm systems. Preprint arXiv:1608.00695. https://arxiv.org/pdf/1608.00695
121. *Flora robotica*. (2017). Project website. http://www.florarobotica.eu
122. Floreano, D., & Mattiussi, C. (2008). *Bio-inspired artificial intelligence: Theories, methods, and technologies*. Cambridge, MA: MIT Press.
123. Fokker, A. D. (1914). Die mittlere Energie rotierender elektrischer Dipole im Strahlungsfeld. *Annalen der Physik, 348*(5), 810–820.
124. Ford, L. R. Jr, & Fulkerson, D. R. (2015). *Flows in networks*. Princeton, NJ: Princeton University Press.
125. Francesca, G., Brambilla, M., Brutschy, A., Garattoni, L., Miletitch, R., Podevijn, G., et al. (2014). An experiment in automatic design of robot swarms: Automode-vanilla, evostick, and human experts. In M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. M. de Oca, C. Solnon, & T. Stützle (Eds.), *Ninth International Conference on Swarm Intelligence (ANTS 2014). Lecture notes in computer science* (Vol. 8667, pp. 25–37).
126. Franks, N. R., Dornhaus, A., Fitzsimmons, J. P., & Stevens, M. (2003). Speed versus accuracy in collective decision making. *Proceedings of the Royal Society of London - Series B: Biological Sciences, 270*, 2457–2463.
127. Franks, N. R., & Sendova-Franks, A. B. (1992). Brood sorting by ants: Distributing the workload over the work-surface. *Behavioral Ecology and Sociobiology, 30*(2), 109–123.
128. Franks, N. R., Wilby, A., Silverman, B. W., & Tofts, C. (1992). Self-organizing nest construction in ants: Sophisticated building by blind bulldozing. *Animal Behaviour, 44*, 357–375.
129. Freeman, P. R. (1983). The secretary problem and its extensions: A review. *International Statistical Review/Revue Internationale de Statistique, 51*(2), 189–206.
130. Frei, R., & Serugendo, G. D. M. (2012). The future of complexity engineering. *Centreal European Journal of Engineering, 2*(2), 164–188. http://dx.doi.org/10.2478/s13531-011-0071-0

131. Galam, S. (1997). Rational group decision making: A random field Ising model at T=0. *Physica A, 238*(1–4), 66–80.
132. Galam, S. (2004). Contrarian deterministic effect on opinion dynamics: The "hung elections scenario". *Physica A, 333*(1), 453–460. http://dx.doi.org/10.1016/j.physa.2003.10.041
133. Galam, S. (2008). Sociophysics: A review of Galam models. *International Journal of Modern Physics C, 19*(3), 409–440.
134. Galam, S., & Moscovici, S. (1991). Towards a theory of collective phenomena: Consensus and attitude changes in groups. *European Journal of Social Psychology, 21*(1), 49–74. https://doi.org/10.1002/ejsp.2420210105
135. Galam, S., & Moscovici, S. (1994). Towards a theory of collective phenomena. II: Conformity and power. *European Journal of Social Psychology, 24*(4), 481–495.
136. Galam, S., & Moscovici, S. (1995). Towards a theory of collective phenomena. III: Conflicts and forms of power. *European Journal of Social Psychology, 25*(2), 217–229.
137. Garnier, S., Gautrais, J., Asadpour, M., Jost, C., & Theraulaz, G. (2009). Self-organized aggregation triggers collective decision making in a group of cockroach-like robots. *Adaptive Behavior, 17*(2), 109–133.
138. Garnier, S., Murphy, T., Lutz, M., Hurme, E., Leblanc, S., & Couzin, I. D. (2013). Stability and responsiveness in a self-organized living architecture. *PLoS Computational Biology, 9*(3), e1002984. https://doi.org/10.1371/journal.pcbi.1002984.
139. Gates, B. (2007). A robot in every home. *Scientific American, 296*(1), 58–65.
140. Gauci, M., Nagpal, R., & Rubenstein, M. (2016). Programmable self-disassembly for shape formation in large-scale robot collectives. In *13th International Symposium on Distributed Autonomous Robotic Systems (DARS 16)*.
141. Gauci, M., Ortiz, M. E., Rubenstein, M., & Nagpal, R. (2017). Error cascades in collective behavior: A case study of the gradient algorithm on 1000 physical agents. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (pp. 1404–1412). International Foundation for Autonomous Agents and Multiagent Systems.
142. Gerkey, B., Vaughan, R. T., & Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003)* (pp. 317–323).
143. Gerling, V., & von Mammen, S. (2016). Robotics for self-organised construction. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, September 2016 (pp. 162–167). https://doi.org/10.1109/FAS-W.2016.45
144. Gierer, A., & Meinhardt, H. (1972). A theory of biological pattern formation. *Biological Cybernetics, 12*(1), 30–39. http://dx.doi.org/10.1007/BF00289234
145. Gjondrekaj, E., Loreti, M., Pugliese, R., Tiezzi, F., Pinciroli, C., Brambilla, M., et al. (2012). Towards a formal verification methodology for collective robotic systems. In *Formal methods and software engineering* (pp. 54–70). Berlin: Springer.
146. Goldstein, I., & Pauzner, A. (2005). Demand–deposit contracts and the probability of bank runs. *The Journal of Finance, 60*(3), 1293–1327.
147. Gomes, J., Mariano, P., & Christensen, A. L. (2015). Cooperative coevolution of partially heterogeneous multiagent systems. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (pp. 297–305). International Foundation for Autonomous Agents and Multiagent Systems.
148. Gomes, J., Urbano, P., & Christensen, A. L. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intelligence, 7*(2–3), 115–144.
149. Gordon, D. M. (1996). The organization of work in social insect colonies. *Nature, 380*, 121–124. https://doi.org/10.1038/380121a0
150. Graham, R., Knuth, D., & Patashnik, O. (1998). *Concrete mathematics: A foundation for computer science*. Reading, MA: Addison-Wesley. ISBN 0-201-55802-5.

151. Grassé, P.-P. (1959). La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux, 6*, 41–83.
152. Gribovskiy, A., Halloy, J., Deneubourg, J.-L., Bleuler, H., & Mondada, F. (2010). Towards mixed societies of chickens and robots. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4722–4728). New York: IEEE.
153. Grimmett, G. (1999). *Percolation*. *Grundlehren der mathematischen Wissenschaften* (Vol. 321). Berlin: Springer.
154. Groß, R., & Dorigo, M. (2008). Evolution of solitary and group transport behaviors for autonomous robots capable of self-assembling. *Adaptive Behavior, 16*(5), 285–305.
155. Groß, R., Magnenat, S., & Mondada, F. (2009). Segregation in swarms of mobile robots based on the Brazil nut effect. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)* (pp. 4349–4356). New York: IEEE.
156. Gross, T., & Sayama, H. (2009). *Adaptive networks: Theory, models, and data*. Berlin: Springer.
157. Guillermo, M. (2005). Morphogens and synaptogenesis in drosophila. *Journal of Neurobiology, 64*(4), 417–434. http://dx.doi.org/10.1002/neu.20165
158. Gunther, N. J. (1993). A simple capacity model of massively parallel transaction systems. In *CMG National Conference* (pp. 1035–1044).
159. Gunther, N. J., Puglia, P., & Tomasette, K. (2015). Hadoop super-linear scalability: The perpetual motion of parallel performance. *ACM Queue, 13*(5), 46–55.
160. Habibi, G., Xie, W., Jellins, M., & McLurkin, J. (2016). Distributed path planning for collective transport using homogeneous multi-robot systems. In N.-Y. Chong & Y.-J. Cho (Eds.), *Distributed Autonomous Robotic Systems: The 12th International Symposium* (pp. 151–164). Tokyo: Springer. ISBN 978-4-431-55879-8. https://doi.org/10.1007/978-4-431-55879-8_11
161. Haken, H. (1977). *Synergetics - An introduction*. Berlin: Springer.
162. Haken, H. (2004). *Synergetics: Introduction and advanced topics*. Berlin: Springer.
163. Halloy, J., Sempo, G., Caprari, G., Rivault, C., Asadpour, M., Tâche, F., et al. (2007). Social integration of robots into groups of cockroaches to control self-organized choices. *Science, 318*(5853), 1155–1158. http://dx.doi.org/10.1126/science.1144259
164. Hamann, H. (2006). *Modeling and Investigation of Robot Swarms*. Master's thesis, University of Stuttgart, Germany.
165. Hamann, H. (2010). *Space-time continuous models of swarm robotics systems: Supporting global-to-local programming*. Berlin: Springer.
166. Hamann, H. (2013). Towards swarm calculus: Urn models of collective decisions and universal properties of swarm performance. *Swarm Intelligence, 7*(2–3), 145–172. http://dx.doi.org/10.1007/s11721-013-0080-0
167. Hamann, H., Divband Soorati, M., Heinrich, M. K., Hofstadler, D. N., Kuksin, I., Veenstra, F., et al. (2017). *Flora robotica* - An architectural system combining living natural plants and distributed robots. Preprint arXiv:1709.04291.
168. Hamann, H., Meyer, B., Schmickl, T., & Crailsheim, K. (2010). A model of symmetry breaking in collective decision-making. In S. Doncieux, B. Girard, A. Guillot, J. Hallam, J.-A. Meyer, & J.-B. Mouret (Eds.), *From animals to animats 11. Lecture notes in artificial intelligence* (Vol. 6226, pp. 639–648), Berlin: Springer. http://dx.doi.org/10.1007/978-3-642-15193-4_60
169. Hamann, H., Schmickl, T., & Crailsheim, K. (2012). A hormone-based controller for evaluation-minimal evolution in decentrally controlled systems. *Artificial Life, 18*(2), 165–198. http://dx.doi.org/10.1162/artl_a_00058
170. Hamann, H., Schmickl, T., & Crailsheim, K. (2012). Self-organized pattern formation in a swarm system as a transient phenomenon of non-linear dynamics. *Mathematical and Computer Modelling of Dynamical Systems, 18*(1), 39–50. http://www.tandfonline.com/doi/abs/10.1080/13873954.2011.601418

171. Hamann, H., Schmickl, T., Wörn, H., & Crailsheim, K. (2012). Analysis of emergent symmetry breaking in collective decision making. *Neural Computing & Applications, 21*(2), 207–218. http://dx.doi.org/10.1007/s00521-010-0368-6
172. Hamann, H., Stradner, J., Schmickl, T., & Crailsheim, K. (2010). A hormone-based controller for evolutionary multi-modular robotics: From single modules to gait learning. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'10)* (pp. 244–251).
173. Hamann, H., Szymanski, M., & Wörn, H. (2007). Orientation in a trail network by exploiting its geometry for swarm robotics. In Y. Shi & M. Dorigo (Eds.), *IEEE Swarm Intelligence Symposium, Honolulu, USA, April 1–5*, Los Alamitos, CA, April 2007 (pp. 310–315). New York: IEEE Press.
174. Hamann, H., Wahby, M., Schmickl, T., Zahadat, P., Hofstadler, D., Støy, K., et al. (2015). *Flora robotica* – Mixed societies of symbiotic robot-plant bio-hybrids. In *Proceedings of IEEE Symposium on Computational Intelligence (IEEE SSCI 2015)* (pp. 1102–1109). New York: IEEE. http://dx.doi.org/10.1109/SSCI.2015.158
175. Hamann, H., & Wörn, H. (2007). An analytical and spatial model of foraging in a swarm of robots. In E. Şahin, W. Spears, & A. F. T. Winfield (Eds.), *Swarm Robotics - Second SAB 2006 International Workshop*. *Lecture notes in computer science* (Vol. 4433, pp. 43–55). Berlin/Heidelberg: Springer.
176. Hamann, H., & Wörn, H. (2008). A framework of space-time continuous models for algorithm design in swarm robotics. *Swarm Intelligence, 2*(2–4), 209–239. http://dx.doi.org/10.1007/s11721-008-0015-3
177. Hansell, M. H. (1984). *Animal architecture and building behaviour*. London: Longman.
178. Harada, K., Corradi, P., Popesku, S., & Liedke, J. (2010). Heterogeneous multi-robot systems. In P. Levi & S. Kernbach (Eds.), *Symbiotic multi-robot organisms: Reliability, adaptability, evolution*. *Cognitive systems monographs* (Vol. 7, pp. 79–163). Berlin: Springer.
179. Harriott, C. E., Seiffert, A. E., Hayes, S. T., & Adams, J. A. (2014). Biologically-inspired human-swarm interaction metrics. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 58*(1), 1471–1475. https://doi.org/10.1177/1541931214581307
180. Hasegawa, E., Mizumoto, N., Kobayashi, K., Dobata, S., Yoshimura, J., Watanabe, S., et al. (2017). Nature of collective decision-making by simple yes/no decision units. *Scientific Reports, 7*(1), 14436. https://doi.org/10.1038/s41598-017-14626-z
181. Hawking, S., & Israel, W. (1987). *300 years of gravitation*. Cambridge: Cambridge University Press.
182. Hayes, A. T. (2002). How many robots? group size and efficiency in collective search tasks. In H. Asama, T. Arai, T. Fukuda, & T. Hasegawa (Eds.), *Distributed autonomous robotic systems 5* (pp. 289–298). Tokyo: Springer. ISBN 978-4-431-65941-9. http://dx.doi.org/10.1007/978-4-431-65941-9_29
183. Hegselmann, R., & Krause, U. (2002). Opinion dynamics and bounded confidence models, analysis, and simulation. *Journal of Artifical Societies and Social Simulation, 5*(3), 1–24.
184. Heinrich, M. K., Wahby, M., Soorati, M. D., Hofstadler, D. N., Zahadat, P., Ayres, P., et al. (2016). Self-organized construction with continuous building material: Higher flexibility based on braided structures. In *Proceedings of the 1st International Workshop on Self-Organising Construction (SOCO)* (pp. 154–159). New York: IEEE. https://doi.org/10.1109/FAS-W.2016.43
185. Helbing, D., Keltsch, J., & Molnár, P. (1997). Modelling the evolution of human trail systems. *Nature, 388*, 47–50.
186. Helbing, D., Schweitzer, F., Keltsch, J., & Molnár, P. (1997). Active walker model for the formation of human and animal trail systems. *Physical Review E, 56*(3), 2527–2539.
187. Hereford, J. M. (2011). Analysis of BEECLUST swarm algorithm. In *Proceedings of the IEEE Symposium on Swarm Intelligence (SIS 2011)* (pp. 192–198). New York: IEEE.
188. Hoddell, S., Melhuish, C., & Holland, O. (1998). Collective sorting and segregation in robots with minimal sensing. In *5th International Conference on the Simulation of Adaptive Behaviour (SAB)*. Cambridge, MA: MIT Press.

189. Hogg, T. (2006). Coordinating microscopic robots in viscous fluids. *Autonomous Agents and Multi-Agent Systems, 14*(3), 271–305.
190. Holland, J. H. (1998). *Emergence - From chaos to order*. New York: Oxford University Press.
191. Huepe, C., Zschaler, G., Do, A.-L., & Gross, T. (2011). Adaptive-network models of swarm dynamics. *New Journal of Physics, 13*, 073022. http://dx.doi.org/10.1088/1367-2630/13/7/073022
192. Ijspeert, A. J., Martinoli, A., Billard, A., & Gambardella, L. M. (2001). Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots, 11*, 149–171. ISSN 0929-5593. https://doi.org/10.1023/A:1011227210047.
193. Ingham, A. G., Levinger, G., Graves, J., & Peckham, V. (1974). The Ringelmann effect: Studies of group size and group performance. *Journal of Experimental Social Psychology, 10*(4), 371–384. ISSN 0022-1031. https://doi.org/10.1016/0022-1031(74)90033-X.
194. Jackson, D. E., Holcombe, M., & Ratnieks, F. L. W. (2004). Trail geometry gives polarity to ant foraging networks. *Nature, 432*, 907–909.
195. Jaeger, J., Surkova, S., Blagov, M., Janssens, H., Kosman, D., Kozlov, K. N., et al. (2004). Dynamic control of positional information in the early *Drosophila* embryo. *Nature, 430*, 368–371. http://dx.doi.org/10.1038/nature02678
196. Jansson, F., Hartley, M., Hinsch, M., Slavkov, I., Carranza, N., Olsson, T. S. G., et al. (2015). Kilombo: A Kilobot simulator to enable effective research in swarm robotics. Preprint arXiv:1511.04285.
197. Jasmine. (2017). Swarm robot - project website. http://www.swarmrobot.org/
198. Jeanne, R. L., & Nordheim, E. V. (1996). Productivity in a social wasp: Per capita output increases with swarm size. *Behavioral Ecology, 7*(1), 43–48.
199. Jeanson, R., Rivault, C., Deneubourg, J.-L., Blanco, S., Fournier, R., Jost, C., et al. (2005). Self-organized aggregation in cockroaches. *Animal Behavior, 69*, 169–180.
200. Johnson, S. (2001). *Emergence: The connected lives of ants, brains, cities, and software*. New York: Scribner.
201. Jones, J. (2010). The emergence and dynamical evolution of complex transport networks from simple low-level behaviours. *International Journal of Unconventional Computing, 6*(2), 125–144.
202. Jones, J. (2010). Characteristics of pattern formation and evolution in approximations of *physarum* transport networks. *Artificial Life, 16*(2), 127–153.
203. Jones, J. L., & Roth, D. (2004). *Robot programming: A practical guide to behavior-based robotics*. New York: McGraw Hill.
204. Kac, M. (1947). Random walk and the theory of Brownian motion. *The American Mathematical Monthly, 54*, 369.
205. Kalthoff, K. (1978). Pattern formation in early insect embryogenesis - data calling for modification of a recent model. *Journal of Cell Science, 29*(1), 1–15.
206. Kanakia, A. P. (2015). *Response Threshold Based Task Allocation in Multi-Agent Systems Performing Concurrent Benefit Tasks with Limited Information*. PhD thesis, University of Colorado Boulder.
207. Kapellmann-Zafra, G., Salomons, N., Kolling, A., & Groß, R. (2016). Human-robot swarm interaction with limited situational awareness. In M. Dorigo (Ed.), *International Conference on Swarm Intelligence (ANTS 2016). Lecture notes in computer science* (pp. 125–136). Berlin: Springer.
208. Karsai, I., & Schmickl, T. (2011). Regulation of task partitioning by a "common stomach": A model of nest construction in social wasps. *Behavioral Ecology, 22*, 819–830. https://doi.org/10.1093/beheco/arr060

209. Kengyel, D., Hamann, H., Zahadat, P., Radspieler, G., Wotawa, F., & Schmickl, T. (2015). Potential of heterogeneity in collective behaviors: A case study on heterogeneous swarms. In Q. Chen, P. Torroni, S. Villata, J. Hsu, & A. Omicini (Eds.), *PRIMA 2015: Principles and practice of multi-agent systems*. *Lecture notes in computer science* (Vol. 9387, pp. 201–217). Berlin: Springer.
210. Kennedy, J., & Eberhart, R. C. (2001). *Swarm intelligence*. Los Altos, CA: Morgan Kaufmann.
211. Kernbach, S., Häbe, D., Kernbach, O., Thenius, R., Radspieler, G., Kimura, T., et al. (2013). Adaptive collective decision-making in limited robot swarms without communication. *The International Journal of Robotics Research, 32*(1), 35–55.
212. Kernbach, S., Thenius, R., Kernbach, O., & Schmickl, T. (2009). Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic swarm. *Adaptive Behavior, 17*, 237–259.
213. Kessler, M. A., & Werner, B. T. (2003). Self-organization of sorted patterned ground. *Science, 299*, 380–383.
214. Kettler, A., Szymanski, M., & Wörn, H. (2012). The Wanda robot and its development system for swarm algorithms. *Advances in autonomous mini robots* (pp. 133–146). Berlin: Springer.
215. Khaluf, Y., Birattari, M., & Hamann, H. (2014). A swarm robotics approach to task allocation under soft deadlines and negligible switching costs. In A. P. del Pobil, E. Chinellato, E. Martinez-Martin, J. Hallam, E. Cervera, & A. Morales (Eds.), *Simulation of adaptive behavior (SAB 2014)*. *Lecture notes in computer science* (Vol. 8575, pp. 270–279). Berlin: Springer.
216. Khaluf, Y., Birattari, M., & Rammig, F. (2013). Probabilistic analysis of long-term swarm performance under spatial interferences. In A.-H. Dediu, C. Martín-Vide, B. Truthe, & M. A. Vega-Rodríguez (Eds.), *Proceedings of Theory and Practice of Natural Computing* (pp. 121–132). Berlin/Heidelberg: Springer. ISBN 978-3-642-45008-2. http://dx.doi.org/10.1007/978-3-642-45008-2_10
217. Kim, L. H., & Follmer, S. (2017). UbiSwarm: Ubiquitous robotic interfaces and investigation of abstract motion as a display. *The Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 1*(3), 66:1–66:20. ISSN 2474-9567. http://doi.acm.org/10.1145/3130931
218. Klein, J. (2003). Continuous 3D agent-based simulations in the breve simulation environment. In *Proceedings of NAACSOS Conference (North American Association for Computational, Social, and Organizational Sciences)*.
219. Kolling, A., Walker, P., Chakraborty, N., Sycara, K., & Lewis, M. (2016). Human interaction with robot swarms: A survey. *IEEE Transactions on Human-Machine Systems, 46*(1), 9–26. ISSN 2168-2291. https://doi.org/10.1109/THMS.2015.2480801
220. König, L., Mostaghim, S., & Schmeck, H. (2009). Decentralized evolution of robotic behavior using finite state machines. *International Journal of Intelligent Computing and Cybernetics, 2*(4), 695–723.
221. Krapivsky, P. L., Redner, S., & Ben-Naim, E. (2010). *A kinetic view of statistical physics*. Cambridge: Cambridge University Press.
222. Kube, C. R., & Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and Autonomous Systems, 30*, 85–101.
223. Kubík, A. (2001). On emergence in evolutionary multiagent systems. In *Proceedings of the 6th European Conference on Artificial Life* (pp. 326–337).
224. Kubík, A. (2003). Toward a formalization of emergence. *Artificial Life, 9*, 41–65.
225. Kuramoto, Y. (1984). *Chemical oscillations, waves, and turbulence*. Berlin: Springer.
226. Kwiatkowska, M., Norman, G., & Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan & S. Qadeer (Eds.), *Proceedings of 23rd International Conference on Computer Aided Verification (CAV'11)*. *Lecture notes in computer science* (Vol. 6806, pp. 585–591). Berlin: Springer.

227. Lazer, D., & Friedman, A. (2007). The network structure of exploration and exploitation. *Administrative Science Quarterly, 52*, 667–694.
228. Le Goc, M., Kim, L. H., Parsaei, A., Fekete, J.-D., Dragicevic, P., & Follmer, S. (2016). Zooids: Building blocks for swarm user interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (pp. 97–109). New York: ACM.
229. Lee, R. E. Jr. (1980). Aggregation of lady beetles on the shores of lakes (coleoptera: Coccinellidae). *American Midland Naturalist, 104*(2), 295–304.
230. Lehman, J., & Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In S. Bullock, J. Noble, R. Watson, & M. A. Bedau (Eds.), *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems* (pp. 329–336). Cambridge, MA: MIT Press.
231. Lehman, J., & Stanley, K. O. (2011). Improving evolvability through novelty search and self-adaptation. In *Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC'11)* (pp. 2693–2700). New York: IEEE.
232. Lemaire, T., Alami, R., & Lacroix, S. (2004). A distributed tasks allocation scheme in multi-UAV context. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04)* (Vol. 4, pp. 3622–3627). New York: IEEE Press. https://doi.org/10.1109/ROBOT.2004.1308816
233. Lenaghan, S. C., Wang, Y., Xi, N., Fukuda, T., Tarn, T., Hamel, W. R., et al. (2013). Grand challenges in bioengineered nanorobotics for cancer therapy. *IEEE Transactions on Biomedical Engineering, 60*(3), 667–673.
234. Leoncini, I., & Rivault, C. (2005). Could species segregation be a consequence of aggregation processes? Example of *Periplaneta americana* (L.) and *P. fuliginosa* (serville). *Ethology, 111*(5), 527–540.
235. Lerman, K. (2004). A model of adaptation in collaborative multi-agent systems. *Adaptive Behavior, 12*(3–4), 187–198.
236. Lerman, K., & Galstyan, A. (2002). Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots, 13*, 127–141.
237. Lerman, K., Galstyan, A., Martinoli, A., & Ijspeert, A. (2001). A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life, 7*, 375–393.
238. Lerman, K., Jones, C., Galstyan, A., & Matarić, M. J. (2006). Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research, 25*(3), 225–241.
239. Lerman, K., Martinoli, A., & Galstyan, A. (2005). A review of probabilistic macroscopic models for swarm robotic systems. In E. Şahin & W. M. Spears (Eds.), *Swarm Robotics - SAB 2004 International Workshop*. *Lecture notes in computer science* (Vol. 3342, pp. 143–152). Berlin: Springer.
240. Levi, P., & Kernbach, S. (Eds.). (2010). *Symbiotic multi-robot organisms: Reliability, adaptability, evolution*. Berlin: Springer.
241. Lind, J. (2000). Issues in agent-oriented software engineering. In M. Wooldridge (Ed.), *Agent-oriented software engineering*. Berlin/Heidelberg/New York: Springer.
242. Ludwig, L., & Gini, M. (2006). Robotic swarm dispersion using wireless intensity signals. In *Distributed autonomous robotic systems 7* (pp. 135–144). Berlin: Springer.
243. Luke, S., Cioffi-Revilla, C., Panait, L., & Sullivan, K. (2004). Mason: A new multi-agent simulation toolkit. In *Proceedings of the 2004 Swarmfest Workshop* (Vol. 8, p. 44).
244. Mack, C. A. (2011). Fifty years of Moore's law. *IEEE Transactions on Semiconductor Manufacturing, 24*(2), 202–207.
245. Madhavan, R., Fregene, K., & Parker, L. E. (2004). Terrain aided distributed heterogeneous multirobot localization and mapping. *Autonomous Robots, 17*, 23–39.
246. Mahmoud, H. (2008). *Pólya urn models*. Boca Raton, FL: Chapman and Hall/CRC.
247. Mallon, E. B., & Franks, N. R. (2000). Ants estimate area using Buffon's needle. *Proceedings of the Royal Society of London B, 267*(1445), 765–770.
248. Mandelbrot, B. B., & Pignoni, R. (1983). *The fractal geometry of nature* (Vol. 173). New York: WH Freeman.

249. Mariano, P., Salem, Z., Mills, R., Zahadat, P., Correia, L., & Schmickl, T. (2017). Design choices for adapting bio-hybrid systems with evolutionary computation. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, New York, NY, USA (pp. 211–212). New York: ACM. ISBN 978-1-4503-4939-0. http://doi.acm.org/10.1145/3067695.3076044
250. Marsili, M., & Zhang, Y.-C. (1998). Interacting individuals leading to Zipf's law. *Physical Review Letters, 80*(12), 2741.
251. Martinoli, A. (1999). *Swarm Intelligence in Autonomous Collective Robotics: From Tools to the Analysis and Synthesis of Distributed Control Strategies*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne.
252. Martinoli, A., Easton, K., & Agassounon, W. (2004). Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *International Journal of Robotics Research, 23*(4), 415–436.
253. Matarić, M. J. (1992). Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation, 8*(3), 304–312.
254. Matarić, M. J. (1992). Minimizing complexity in controlling a mobile robot population. In *IEEE International Conference on Robotics and Automation* (pp. 830–835). New York: IEEE.
255. Matarić, M. J. (1993). Designing emergent behaviors: From local interactions to collective intelligence. *Proceedings of the Second International Conference on From Animals to Animats 2: Simulation of Adaptive Behavior* (pp. 432–441).
256. Matarić, M. J. (1995). Designing and understanding adaptive group behavior. *Adaptive Behavior, 4*, 51–80.
257. Matarić, M. J. (1997). Reinforcement learning in the multi-robot domain. *Autonomous Robots, 4*(1), 73–83.
258. Mathews, N., Christensen, A. L., Ferrante, E., O'Grady, R., & Dorigo, M. (2010). Establishing spatially targeted communication in a heterogeneous robot swarm. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (pp. 939–946). International Foundation for Autonomous Agents and Multiagent Systems.
259. Mayet, R., Roberz, J., Schmickl, T., & Crailsheim, K. (2010). Antbots: A feasible visual emulation of pheromone trails for swarm robots. In M. Dorigo, M. Birattari, G. A. Di Caro, R. Doursat, A. P. Engelbrecht, D. Floreano, L. M. Gambardella, R. Groß, E. Şahin, H. Sayama, & T. Stützle (Eds.), *Swarm Intelligence: 7th International Conference, ANTS 2010. Lecture notes in computer science* (Vol. 6234, pp. 84–94). Berlin/Heidelberg/New York: Springer. ISBN 978-3-642-15460-7. https://doi.org/10.1007/978-3-642-15461-4
260. McEvoy, M. A., & Correll, N. (2015). Materials that couple sensing, actuation, computation, and communication. *Science, 347*(6228), 1261689 . ISSN 0036-8075. https://doi.org/10.1126/science.1261689.
261. McLurkin, J., Lynch, A. J., Rixner, S., Barr, T. W., Chou, A., Foster, K., et al. (2013). A low-cost multi-robot system for research, teaching, and outreach. In *Distributed autonomous robotic systems* (pp. 597–609). Berlin: Springer.
262. McLurkin, J., & Smith, J. (2004). Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *Distributed Autonomous Robotic Systems Conference*.
263. McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., & Schmidt, B. (2006). Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team has Gone Before* (pp. 72–75).
264. Meinhardt, H. (1982). *Models of biological pattern formation*. New York: Academic.
265. Meinhardt, H. (2003). *The algorithmic beauty of sea shells*. Berlin: Springer.
266. Meinhardt, H., & Gierer, A. (1974). Applications of a theory of biological pattern formation based on lateral inhibition. *Journal of Cell Science, 15*(2), 321–346. ISSN 0021-9533. http://view.ncbi.nlm.nih.gov/pubmed/4859215
267. Meinhardt, H., & Gierer, A. (2000). Pattern formation by local self-activation and lateral inhibition. *Bioessays, 22*, 753–760.

268. Meinhardt, H., & Klingler, M. (1987). A model for pattern formation on the shells of molluscs. *Journal of Theoretical Biology, 126*, 63–69.
269. Meister, T., Thenius, R., Kengyel, D., & Schmickl, T. (2013). Cooperation of two different swarms controlled by BEECLUST algorithm. In *Mathematical models for the living systems and life sciences (ECAL)* (pp. 1124–1125).
270. Melhuish, C., Wilson, M., & Sendova-Franks, A. (2001). Patch sorting: Multi-object clustering using minimalist robots. In J. Kelemen & P. Sosík (Eds.), *Advances in Artificial Life: 6th European Conference, ECAL 2001 Prague, Czech Republic, September 10–14, 2001 Proceedings* (pp. 543–552). Berlin/Heidelberg: Springer. ISBN 978-3-540-44811-2. https://doi.org/10.1007/3-540-44811-X_62
271. Mellinger, D., Shomin, M., Michael, N., & Kumar, V. (2013). Cooperative grasping and transport using multiple quadrotors. In *Distributed autonomous robotic systems* (pp. 545–558). Berlin: Springer.
272. Merkle, D., Middendorf, M., & Scheidler, A. (2007). Swarm controlled emergence-designing an anti-clustering ant system. In *IEEE Swarm Intelligence Symposium* (pp. 242–249). New York: IEEE.
273. Merkle, D., Middendorf, M., & Scheidler, A. (2008). Organic computing and swarm intelligence. In C. Blum & D. Merkle (Eds.), *Swarm intelligence: Introduction and applications*. Berlin: Springer.
274. Meyer, B., Beekman, M., & Dussutour, A. (2008). Noise-induced adaptive decision-making in ant-foraging. In *Simulation of adaptive behavior (SAB). Lecture notes in computer science* (Vol. 5040, pp. 415–425). Berlin: Springer.
275. Meyer, B., Renner, C., & Maehle, E. (2016). Versatile sensor and communication expansion set for the autonomous underwater vehicle MONSUN. In *Advances in Cooperative Robotics: Proceedings of the 19th International Conference on CLAWAR 2016* (pp. 250–257). Singapore: World Scientific.
276. Mill, J. S. (1843). *A system of logic: Ratiocinative and inductive*. London: John W. Parker and Son.
277. Milutinovic, D., & Lima, P. (2006). Modeling and optimal centralized control of a large-size robotic population. *IEEE Transactions on Robotics, 22*(6), 1280–1285.
278. Milutinovic, D., & Lima, P. (2007). *Cells and robots: Modeling and control of large-size agent populations*. Berlin: Springer.
279. Misner, C. W., Thorne, K. S., & Wheeler, J. A. (2017). *Gravitation*. Princeton, NJ: Princeton University Press.
280. Möbius, M. E., Lauderdale, B. E., Nagel, S. R., & Jaeger, H. M. (2001). Brazil-nut effect: Size separation of granular particles. *Nature, 414*(6861), 270.
281. Moeslinger, C., Schmickl, T., & Crailsheim, K. (2010). Emergent flocking with low-end swarm robots. In M. Dorigo, M. Birattari, G. Di Caro, R. Doursat, A. Engelbrecht, D. Floreano, L. Gambardella, R. Groß, E. Sahin, H. Sayama, & T. Stützle (Eds.), *Swarm intelligence. Lecture notes in computer science* (Vol. 6234, pp. 424–431). Berlin/Heidelberg: Springer.
282. Moeslinger, C., Schmickl, T., & Crailsheim, K. (2011). A minimalist flocking algorithm for swarm robots. In *Advances in Artificial Life: Darwin Meets von Neumann (ECAL'09). Lecture notes in computer science* (Vol. 5778, pp. 357–382). Heidelberg/Berlin: Springer.
283. Moioli, R., Vargas, P. A., & Husbands, P. (2010). Exploring the Kuramoto model of coupled oscillators in minimally cognitive evolutionary robotics tasks. In *WCCI 2010 IEEE World Congress on Computational Intelligence - CEC IEEE* (pp. 2483–2490).
284. Monajjemi, V. M., Wawerla, J., Vaughan, R., & Mori, G. (2013). HRI in the sky: Creating and commanding teams of UAVs with a vision-mediated gestural interface. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013 (pp. 617–623). https://doi.org/10.1109/IROS.2013.6696415
285. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., et al. (2009). The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions* (Vol. 1, pp. 59–65).

286. Mondada, F., Franzi, E., & Ienne, P. (1994). Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Proceedings of the 3rd International Symposium on Experimental Robotics* (pp. 501–513). Berlin: Springer.
287. Mondada, L., Karim, M. E., & Mondada, F. (2016). Electroencephalography as implicit communication channel for proximal interaction between humans and robot swarms. *Swarm Intelligence, 10*(4), 247–265. ISSN 1935-3820. https://doi.org/10.1007/s11721-016-0127-0
288. Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I., Floreano, D., Deneubourg, J.-L., et al. (2004). SWARM-BOT: A new distributed robotic concept. *Autonomous Robots, Special Issue on Swarm Robotics, 17*(2–3), 193–221. doi: NA.
289. Murray, J. D. (1981). A prepattern formation mechanism for animal coat markings. *Journal of Theoretical Biology, 88*, 161–199.
290. Murray, J. D. (2003). On the mechanochemical theory of biological pattern formation with application to vasculogenesis. *Comptes Rendus Biologies, 326*(2), 239–252.
291. Nagi, J., Giusti, A., Gambardella, L. M., & Di Caro, G. A. (2014). Human-swarm interaction using spatial gestures. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2014 (pp. 3834–3841). https://doi.org/10.1109/IROS.2014.6943101
292. Nair, R., Ito, T., Tambe, M., & Marsella, S. (2002). Task allocation in the RoboCup rescue simulation domain: A short note. In A. Birk, S. Coradeschi, & S. Tadokoro (Eds.), *RoboCup 2001: Robot Soccer World Cup V* (Vol. 2377, pp. 1–22). Berlin/Heidelberg: Springer. http://dx.doi.org/10.1007/3-540-45603-1_129
293. Nolfi, S., Bongard, J., Husbands, P., & Floreano, D. (2016). Evolutionary robotics. In *Springer handbook of robotics* (pp. 2035–2068). Berlin: Springer.
294. Nouyan, S., Campo, A., & Dorigo, M. (2008). Path formation in a robot swarm: Self-organized strategies to find your way home. *Swarm Intelligence, 2*(1), 1–23.
295. Nouyan, S., Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2009). Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation, 13*(4), 695–711.
296. O'Grady, R., Groß, R., Mondada, F., Bonani, M., & Dorigo, M. (2005). Self-assembly on demand in a group of physical autonomous mobile robots navigating rough terrain. In *Advances in Artificial Life, 8th European Conference (ECAL)* (pp. 272–281). Berlin: Springer.
297. O'Keeffe, K. P., Hong, H., & Strogatz, S. H. (2017). Oscillators that sync and swarm. *Nature Communications, 8*, 1504.
298. Olfati-Saber, R., Fax, A., & Murray, R. M. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE, 95*(1), 215–233.
299. Østergaard, E. H., Sukhatme, G. S., & Matarić, M. J. (2001). Emergent bucket brigading: A simple mechanisms for improving performance in multi-robot constrained-space foraging tasks. In E. André, S. Sen, C. Frasson, & J. P. Müller (Eds.), *Proceedings of the Fifth International Conference on Autonomous Agents (AGENTS'01)*, New York, NY, USA (pp. 29–35). New York: ACM. ISBN 1-58113-326-X. http://doi.acm.org/10.1145/375735.375825
300. Osterloh, C., Pionteck, T., & Maehle, E. (2012). MONSUN II: A small and inexpensive AUV for underwater swarms. In *ROBOTIK 2012 - 7th German Conference on Robotics*.
301. Ostwald, W. (1897). Studien über die Bildung und Umwandlung fester Körper. *Zeitschrift für physikalische Chemie, 22*(1), 289–330.
302. Parrish, J. K., & Edelstein-Keshet, L. (1999). Complexity, pattern, and evolutionary trade-offs in animal aggregation. *Science, 284*(5411), 99–101. ISSN 0036-8075. https://doi.org/10.1126/science.284.5411.99
303. Parunak, H. V. D., & Brueckner, S. A. (2004). Engineering swarming systems. In *Methodologies and software engineering for agent systems* (pp. 341–376). Berlin: Springer.
304. Payton, D., Daily, M., Estowski, R., Howard, M., & Lee, C. (2001). Pheromone robotics. *Autonomous Robots, 11*(3), 319–324.
305. Peires, F. T. (1926). Tensile tests for cotton yarns. *Journal of the Textile Institute, 17*, 355–368.
306. Petersen, K., Nagpal, R., & Werfel, J. (2011). TERMES: An autonomous robotic system for three-dimensional collective construction. *Proceedings Robotics: Science & Systems VII* (pp. 257–264).

307. Pickem, D., Lee, M., & Egerstedt, M. (2015). The GRITSBot in its natural habitat: A multi-robot testbed. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4062–4067). New York: IEEE.
308. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., et al. (2012). ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence, 6*(4), 271–295. ISSN 1935-3812. http://dx.doi.org/10.1007/s11721-012-0072-5
309. Pini, G., Brutschy, A., Francesca, G., Dorigo, M., & Birattari, M. (2012). Multi-armed bandit formulation of the task partitioning problem in swarm robotics. In *8th International Conference on Swarm Intelligence (ANTS)* (pp. 109–120). Berlin: Springer.
310. Planck, M. (1917). Über einen Satz der statistischen Dynamik und seine Erweiterung in der Quantentheorie. *Sitzungsberichte der Preußischen Akademie der Wissenschaften, 24*, 324–341.
311. Podevijn, G., O'Grady, R., Mathews, N., Gilles, A., Fantini-Hauwel, C., & Dorigo, M. (2016). Investigating the effect of increasing robot group sizes on the human psychophysiological state in the context of human–swarm interaction. *Swarm Intelligence, 10*(3), 1–18. ISSN 1935-3820. http://dx.doi.org/10.1007/s11721-016-0124-3
312. Pólya, G., & Eggenberger, F. (1923). Über die Statistik verketteter Vorgänge. *Zeitschrift für Angewandte Mathematik und Mechanik, 3*(4), 279–289.
313. Popkin, G. (2016). The physics of life. *Nature, 529*, 16–18. https://doi.org/10.1038/529016a
314. Potter, M. A., Meeden, L. A., & Schultz, A. C. (2001). Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 1337–1343). Los Altos, CA: Morgan Kaufmann.
315. Pourmehr, S., Monajjemi, V. M., Vaughan, R., & Mori, G. (2013). "you two! Take off!": Creating, modifying and commanding groups of robots using face engagement and indirect speech in voice commands. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013 (pp. 137–142). https://doi.org/10.1109/IROS.2013.6696344
316. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2002). *Numerical recipes in C++*. Cambridge: Cambridge University Press.
317. Prigogine, I. (1997). *The end of certainty: Time, chaos, and the new laws of nature*. New York: Free Press.
318. Prorok, A., Ani Hsieh, M., & Kumar, V. (2015). Fast redistribution of a swarm of heterogeneous robots. In *International Conference on Bio-inspired Information and Communications Technologies (BICT)*.
319. Prorok, A., Ani Hsieh, M., & Kumar, V. (2016). Formalizing the impact of diversity on performance in a heterogeneous swarm of robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5364–5371). https://doi.org/10.1109/ICRA.2016.7487748
320. Prorok, A., Correll, N., & Martinoli, A. (2011). Multi-level spatial models for swarm-robotic systems. *The International Journal of Robotics Research, 30*(5), 574–589.
321. Raischel, F., Kun, F., & Herrmann, H. J. (2006). Fiber bundle models for composite materials. In *Conference on Damage in Composite Materials*.
322. Ramaley, J. F. (1969). Buffon's noodle problem. *The American Mathematical Monthly, 76*(8), 916–918. http://www.jstor.org/stable/2317945
323. Ratnieks, F. L. W., & Anderson, C. (1999). Task partitioning in insect societies. *Insectes Sociaux, 46*(2), 95–108. https://doi.org/10.1007/s000400050119
324. Reina, A., Dorigo, M., & Trianni, V. (2014). Towards a cognitive design pattern for collective decision-making. In M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. M. de Oca, C. Solnon, & T. Stützle (Eds.), *Swarm intelligence*. *Lecture notes in computer science* (Vol. 8667, pp. 194–205). Berlin: Springer International Publishing. ISBN 978-3-319-09951-4. http://dx.doi.org/10.1007/978-3-319-09952-1_17
325. Reina, A., Valentini, G., Fernández-Oto, C., Dorigo, M., & Trianni, V. (2015). A design pattern for decentralised decision making. *PLoS One, 10*(10), 1–18. https://doi.org/10.1371/journal.pone.0140950

326. Resnick, M. (1994). *Turtles, termites, and traffic jams*. Cambridge, MA: MIT Press.
327. Reynolds, C. W. (1987). Flocks, herds, and schools. *Computer Graphics, 21*(4), 25–34.
328. Riedo, F., Chevalier, M., Magnenat, S., & Mondada, F. (2013). Thymio II, a robot that grows wiser with children. In *IEEE Workshop on Advanced Robotics and Its Social Impacts (ARSO 2013)* (pp. 187–193). New York: IEEE.
329. Risken, H. (1984). *The Fokker-Planck equation*. Berlin: Springer.
330. Rosenblueth, A., Wiener, N., & Bigelow, J. (1943). Behavior, purpose and teleology. *Philosophy of Science, 10*(1), 18–24. ISSN 00318248, 1539767X. http://www.jstor.org/stable/184878
331. Rubenstein, M., Ahler, C., & Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. In *IEEE International Conference on Robotics and Automation (ICRA 2012)* (pp. 3293–3298). https://doi.org/10.1109/ICRA.2012.6224638
332. Rubenstein, M., Cornejo, A., & Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. *Science, 345*(6198), 795–799. http://dx.doi.org/10.1126/science.1254295
333. Rubenstein, M., & Shen, W.-M. (2009). Scalable self-assembly and self-repair in a collective of robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 2009.
334. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536.
335. Russell, B. (1923). Vagueness. *Australasian Journal of Psychology and Philosophy, 1*(2), 84–92. http://dx.doi.org/10.1080/00048402308540623
336. Russell, R. A. (1997). Heat trails as short-lived navigational markers for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation* (Vol. 4, pp. 3534–3539).
337. Russell, S. J., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Englewood, Cliffs, NJ: Prentice Hall.
338. Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In E. Şahin & W. M. Spears (Eds.), *Swarm Robotics - SAB 2004 International Workshop*. *Lecture notes in computer science* (Vol. 3342, pp. 10–20). Berlin: Springer.
339. Savkin, A. V. (2004). Coordinated collective motion of groups of autonomous mobile robots: Analysis of Vicsek's model. *IEEE Transactions on Automatic Control, 49*(6), 981–982.
340. Scheidler, A. (2011). Dynamics of majority rule with differential latencies. *Physical Review E, 83*(3), 031116.
341. Scheidler, A., Merkle, D., & Middendorf, M. (2013). Swarm controlled emergence for ant clustering. *International Journal of Intelligent Computing and Cybernetics, 6*(1), 62–82.
342. Schmelzer, J. W. P. (2006). *Nucleation theory and applications*. New York: Wiley.
343. Schmickl, T., & Crailsheim, K. (2004). Costs of environmental fluctuations and benefits of dynamic decentralized foraging decisions in honey bees. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems, 12*, 263–277.
344. Schmickl, T., & Hamann, H. (2011). BEECLUST: A swarm algorithm derived from honeybees. In Y. Xiao (Ed.), *Bio-inspired computing and communication networks* (pp. 95–137). Boca Raton, FL: CRC Press.
345. Schmickl, T., Hamann, H., & Crailsheim, K. (2011). Modelling a hormone-inspired controller for individual- and multi-modular robotic systems. *Mathematical and Computer Modelling of Dynamical Systems, 17*(3), 221–242.
346. Schmickl, T., Hamann, H., Wörn, H., & Crailsheim, K. (2009). Two different approaches to a macroscopic model of a bio-inspired robotic swarm. *Robotics and Autonomous Systems, 57*(9), 913–921. http://dx.doi.org/10.1016/j.robot.2009.06.002
347. Schmickl, T., Möslinger, C., & Crailsheim, K. (2007). Collective perception in a robot swarm. In E. Şahin, W. M. Spears, & A. F. T. Winfield (Eds.), *Swarm Robotics - Second SAB 2006 International Workshop*. *Lecture notes in computer science* (Vol. 4433). Heidelberg/Berlin: Springer.

348. Schmickl, T., Stradner, J., Hamann, H., Winkler, L., & Crailsheim, K. (2011). Major feedback loops supporting artificial evolution in multi-modular robotics. In S. Doncieux, N. Bredèche, & J.-B. Mouret (Eds.), *New horizons in evolutionary robotics*. *Studies in computational intelligence* (Vol. 341, pp. 195–209). Berlin/Heidelberg: Springer. ISBN 978-3-642-18271-6. https://doi.org/10.1007/978-3-642-18272-3
349. Schmickl, T., Thenius, R., Möslinger, C., Radspieler, G., Kernbach, S., & Crailsheim, K. (2008). Get in touch: Cooperative decision making based on robot-to-robot collisions. *Autonomous Agents and Multi-Agent Systems, 18*(1), 133–155.
350. Schmickl, T., Thenius, R., Moslinger, C., Timmis, J., Tyrrell, A., Read, M., et al. (2011). Cocoro – the self-aware underwater swarm. In *2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops* (pp. 120–126). https://doi.org/10.1109/SASOW.2011.11
351. Schultz, A. C., Grefenstette, J. J., & Adams, W. (1996). Robo-Shepherd: Learning complex robotic behaviors. In M. Jamshidi, F. Pin, & P. Dauchez (Eds.), *Proceedings of the International Symposium on Robotics and Automation (ICRA'96)* (Vol. 6, pp. 763–768). New York, NY: ASME Press.
352. Schumacher, R. (2002). Book review: Achim Stephan: Emergenz. Von der Unvorhersagbarkeit zur Selbstorganisation. *European Journal of Philosophy, 10*(3), 415–419 (Dresden/München: Dresden University Press, 1999).
353. Schweitzer, F. (2002). Brownian agent models for swarm and chemotactic interaction. In D. Polani, J. Kim, & T. Martinetz (Eds.), *Fifth German Workshop on Artificial Life. Abstracting and Synthesizing the Principles of Living Systems* (pp. 181–190). Akademische Verlagsgesellschaft Aka.
354. Schweitzer, F. (2003). *Brownian agents and active particles. On the emergence of complex behavior in the natural and social sciences*. Berlin: Springer.
355. Schweitzer, F., Lao, K., & Family, F. (1997). Active random walkers simulate trunk trail formation by ants. *BioSystems, 41*, 153–166.
356. Schweitzer, F., & Schimansky-Geier, L. (1994). Clustering of active walkers in a two-component system. *Physica A, 206*, 359–379.
357. Sempo, G., Depickère, S., Amé, J.-M., Detrain, C., Halloy, J., & Deneubourg, J.-L. (2006). Integration of an autonomous artificial agent in an insect society: Experimental validation. In S. Nolfi, G. Baldassarre, R. Calabretta, J. C. T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, & D. Parisi (Eds.), *From Animals to Animats 9: 9th International Conference on Simulation of Adaptive Behavior, SAB 2006, Rome, Italy, September 25–29, 2006. Proceedings* (pp. 703–712). Berlin/Heidelberg: Springer. ISBN 978-3-540-38615-5. https://doi.org/10.1007/11840541_58
358. Seyfried, J., Szymanski, M., Bender, N., Estaña, R., Thiel, M., & Wörn, H. (2005). The I-SWARM project: Intelligent small world autonomous robots for micro-manipulation. In E. Şahin & W. M. Spears (Eds.), *Swarm Robotics Workshop: State-of-the-Art Survey* (pp. 70–83). Berlin: Springer.
359. Sharkey, A. J. C. (2007). Swarm robotics and minimalism. *Connection Science, 19*(3), 245–260.
360. Sood, V., & Redner, S. (2005). Voter model on heterogeneous graphs. *Physical Review Letters, 94*(17), 178701.
361. Soysal, O., & Şahin, E. (2007). A macroscopic model for self-organized aggregation in swarm robotic systems. In E. Şahin, W. M. Spears, & A. F. T. Winfield (Eds.), *Swarm Robotics - Second SAB 2006 International Workshop*. *Lecture notes in computer science* (Vol. 4433, pp. 27–42). Berlin: Springer.
362. Springel, V., White, S. D. M., Jenkins, A., Frenk, C. S., Yoshida, N., Gao, L., et al. (2005). Simulations of the formation, evolution and clustering of galaxies and quasars. *Nature, 435*, 629–636. http://www.nature.com/nature/journal/v435/n7042/full/nature03597.html
363. Stephan, A. (1999). *Emergenz: Von der Unvorhersagbarkeit zur Selbstorganisation*. Dresden, Munich: Dresden University Press.

364. Stepney, S., Polack, F., & Turner, H. (2006). Engineering emergence. In *CEC 2006: 11th IEEE International Conference on Engineering of Complex Computer Systems, Stanford, CA, USA*, Los Alamitos, CA, August 2006. New York: IEEE Press.
365. Støy, K., & Nagpal, R. (2004). Self-repair through scale independent self-reconfiguration. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004 (IROS 2004). Proceedings* (Vol. 2, pp. 2062–2067). NewYork: IEEE.
366. Strogatz, S. H. (2001). Exploring complex networks. *Nature, 410*(6825), 268–276. http://www.nature.com/nature/journal/v410/n6825/abs/410268a0.html
367. Strogatz, S. H., Abrams, D. M., McRobie, A., Eckhardt, B., & Ott, E. (2005). Theoretical mechanics: Crowd synchrony on the Millennium Bridge. *Nature, 438*(7064), 43–44.
368. Sugawara, K., Kazama, T., & Watanabe, T. (2004). Foraging behavior of interacting robots with virtual pheromone. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Los Alamitos, CA (pp. 3074–3079). New York: IEEE Press.
369. Sugawara, K., & Sno, M. (1997). Cooperative acceleration of task performance: Foraging behavior of interacting multi-robots system. *Physica D, 100*, 343–354.
370. Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
371. Sznajd-Weron, K., & Sznajd, J. (2000). Opinion evolution in closed community. *International Journal of Modern Physics C, 11*(06), 1157–1165.
372. Szopek, M., Schmickl, T., Thenius, R., Radspieler, G., & Crailsheim, K. (2013). Dynamics of collective decision making of honeybees in complex temperature fields. *PLoS One, 8*(10), e76250. https://doi.org/10.1371/journal.pone.0076250. http://dx.doi.org/10.1371%2Fjournal.pone.0076250
373. Tabony, J., & Job, D. (1992). Gravitational symmetry breaking in microtubular dissipative structures. *Proceedings of the National Academy of Sciences of the United States of America, 89*(15), 6948–6952.
374. Tarapore, D., Christensen, A. L., & Timmis, J. (2017). Generic, scalable and decentralized fault detection for robot swarms. *PLoS One, 12*(8), 1–29. https://doi.org/10.1371/journal.pone.0182058
375. Tarapore, D., Lima, P. U., Carneiro, J., & Christensen, A. L. (2015). To err is robotic, to tolerate immunological: Fault detection in multirobot systems. *Bioinspiration & Biomimetics, 10*(1), 016014.
376. (Nagpal, R., et al.) The Kilobot Project, Self-Organizing Systems Research Group. (2013). Website. https://ssr.seas.harvard.edu/kilobots
377. Theraulaz, G., & Bonabeau, E. (1995). Coordination in distributed building. *Science, 269*, 686–688.
378. Theraulaz, G., & Bonabeau, E. (1995). Modelling the collective building of complex architectures in social insects with lattice swarms. *Journal of Theoretical Biology, 177*, 381–400.
379. Theraulaz, G., Bonabeau, E., Nicolis, S. C., Solé, R. V., Fourcassié, V., Blanco, S., et al. (2002). Spatial patterns in ant colonies. *Proceedings of the National Academy of Sciences of the United States of America, 99*(15), 9645–9649.
380. Thompson, D. W. (1917). *On growth and form: The complete revised edition*. Cambridge: Cambridge University Press.
381. Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. Cambridge, MA: MIT Press.
382. Toner, J., & Tu, Y. (1998). Flocks, herds, and schools: A quantitative theory of flocking. *Physical Review E, 58*(4), 4828–4858.
383. Trianni, V. (2008). *Evolutionary swarm robotics - Evolving self-organising behaviours in groups of autonomous robots*. *Studies in computational intelligence* (Vol. 108). Berlin: Springer.
384. Trianni, V., Groß, R., Labella, T. H., Şahin, E., & Dorigo, M. (2003). Evolving aggregation behaviors in a swarm of robots. In W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, & J. T. Kim (Eds.), *Advances in Artificial Life (ECAL 2003)*. *Lecture notes in artificial intelligence* (Vol. 2801, pp. 865–874). Berlin: Springer.

385. Trianni, V., Ijsselmuiden, J., & Haken, R. (2016). *The SAGA concept: Swarm robotics for agricultural applications*. Technical report. http://laral.istc.cnr.it/saga/wp-content/uploads/2016/09/saga-dars2016.pdf
386. Trianni, V., Labella, T. H., & Dorigo, M. (2004). Evolution of direct communication for a swarm-bot performing hole avoidance. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, & T. Stützle (Eds.), *Ant colony optimization and swarm intelligence (ANTS 2004)*. *Lecture notes in computer science* (Vol. 3172, pp. 130–141). Berlin: Springer.
387. Tuci, E., Groß, R., Trianni, V., Mondada, F., Bonani, M., & Dorigo, M. (2006). Cooperation through self-assembly in multi-robot systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS), 1*(2), 115–150.
388. Turgut, A., Çelikkanat, H., Gökçe, F., & Şahin, E. (2008). Self-organized flocking in mobile robot swarms. *Swarm Intelligence, 2*(2), 97–120. http://dx.doi.org/10.1007/s11721-008-0016-2
389. Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, B237*(641), 37–72.
390. Twu, P., Mostofi, Y., & Egerstedt, M. (2014). A measure of heterogeneity in multi-agent systems. In *American Control Conference* (pp. 3972–3977).
391. Tyrrell, A., Auer, G., & Bettstetter, C. (2006). Fireflies as role models for synchronization in ad hoc networks. In *Proceedings of the 1st International Conference on Bio-inspired Models of Network, Information and Computing Systems*. New York: ACM.
392. Valentini, G. (2017). *Achieving consensus in robot swarms: Design and analysis of strategies for the best-of-n problem*. Berlin: Springer. ISBN 978-3-319-53608-8. https://doi.org/10.1007/978-3-319-53609-5
393. Valentini, G., Brambilla, D., Hamann, H., & Dorigo, M. (2016). Collective perception of environmental features in a robot swarm. In *10th International Conference on Swarm Intelligence, ANTS 2016*. *Lecture notes in computer science* (Vol. 9882, pp. 65–76). Berlin: Springer.
394. Valentini, G., Ferrante, E., & Dorigo, M. (2017). The best-of-n problem in robot swarms: Formalization, state of the art, and novel perspectives. *Frontiers in Robotics and AI, 4*, 9. ISSN 2296-9144. http://journal.frontiersin.org/article/10.3389/frobt.2017.00009
395. Valentini, G., Ferrante, E., Hamann, H., & Dorigo, M. (2016). Collective decision with 100 Kilobots: Speed vs accuracy in binary discrimination problems. *Journal of Autonomous Agents and Multi-Agent Systems, 30*(3), 553–580. http://dx.doi.org/10.1007/s10458-015-9323-3
396. Valentini, G., Hamann, H., & Dorigo, M. (2014). Self-organized collective decision making: The weighted voter model. In A. Lomuscio, P. Scerri, A. Bazzan, & M. Huhns (Eds.), *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*. IFAAMAS.
397. Valentini, G., Hamann, H., & Dorigo, M. (2015). Self-organized collective decisions in a robot swarm. In *AAAI-15 Video Proceedings*. Palo Alto, CA: AAAI Press. https://www.youtube.com/watch?v=5lz_HnOLBW4.
398. Valentini, G., Hamann, H., & Dorigo, M. (2015). Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off. In R. Bordini, E. Elkind, G. Weiss, & P. Yolum (Eds.), *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)* (pp. 1305–1314). IFAAMAS. http://dl.acm.org/citation.cfm?id=2773319.
399. Vestartas, P., Heinrich, M. K., Zwierzycki, M., Leon, D. A., Cheheltan, A., La Magna, R., & Ayres, P. (2018). Design tools and workflows for braided structures. In K. De Rycke, C. Gengnagel, O. Baverel, J. Burry, C. Mueller, M. M. Nguyen, P. Rahm, & M. R. Thomsen (Eds.), *Humanizing Digital Reality: Design Modelling Symposium Paris 2017* (pp. 671–681). Singapore: Springer. ISBN 978-981-10-6611-5. https://doi.org/10.1007/978-981-10-6611-5_55
400. Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., & Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical Review Letters, 6*(75), 1226–1229.

401. Vicsek, T., & Zafeiris, A. (2012). Collective motion. *Physics Reports, 517*(3–4), 71–140.
402. Vigelius, M., Meyer, B., & Pascoe, G. (2014). Multiscale modelling and analysis of collective decision making in swarm robotics. *PLoS One, 9*(11), 1–19. https://doi.org/10.1371/journal.pone.0111542
403. Vinković, D., & Kirman, A. (2006). A physical analogue of the Schelling model. *Proceedings of the National Academy of Sciences of the United States of America, 103*(51), 19261–19265.
404. von Frisch, K. (1974). *Animal architecture*. San Diego, CA: Harcourt.
405. Voorhees, P. W. (1985). The theory of Ostwald ripening. *Journal of Statistical Physics, 38*(1), 231–252.
406. Wahby, M., Weinhold, A., & Hamann, H. (2016). Revisiting BEECLUST: Aggregation of swarm robots with adaptiveness to different light settings. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT 2015)*, pp. 272–279. ICST. http://dx.doi.org/10.4108/eai.3-12-2015.2262877
407. Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature, 393*(6684), 440–442.
408. Webb, B. (2001). Can robots make good models of biological behaviour? *Behavioral and Brain Sciences, 24*, 1033–1050.
409. Webb, B. (2002). Robots in invertebrate neuroscience. *Nature, 417*, 359–363.
410. Webb, B., & Scutt, T. (2000). A simple latency-dependent spiking-neuron model of cricket phonotaxis. *Biological Cybernetics, 82*, 247–269.
411. Weinberg, S. (1995). Reductionism redux. *The New York Review of Books, 42*(15), 5.
412. Wells, H., Wells, P. H., & Cook, P. (1990). The importance of overwinter aggregation for reproductive success of monarch butterflies (*danaus plexippus l.*). *Journal of Theoretical Biology, 147*(1), 115–131. ISSN 0022-5193. http://dx.doi.org/10.1016/S0022-5193(05)80255-3.
413. Werfel, J., Petersen, K., & Nagpal, R. (2014). Designing collective behavior in a termite-inspired robot construction team. *Science, 343*(6172), 754–758. http://dx.doi.org/10.1126/science.1245842
414. Whiteson, S., Kohl, N., Miikkulainen, R., & Stone, P. (2003). Evolving keepaway soccer players through task decomposition. In *Genetic and Evolutionary Computation-GECCO 2003* (pp. 20–212). Berlin: Springer.
415. Whiteson, S., & Stone, P. (2006). Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research, 7*, 877–917.
416. Wilson, S., Gameros, R., Sheely, M., Lin, M., Dover, K., Gevorkyan, R., et al. (2016). Pheeno, a versatile swarm robotic research and education platform. *IEEE Robotics and Automation Letters, 1*(2), 884–891.
417. Wilson, M., Melhuish, C., Sendova-Franks, A. B., & Scholes, S. (2004). Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies. *Autonomous Robots, 17*, 115–136.
418. Wilson, S., Pavlic, T. P., Kumar, G. P., Buffin, A., Pratt, S. C., & Berman, S. (2014). Design of ant-inspired stochastic control policies for collective transport by robotic swarms. *Swarm Intelligence, 8*(4), 303–327.
419. Winfield, A. F. T., Harper, C. J., & Nembrini, J. (2004). Towards dependable swarms and a new discipline of swarm engineering. In *International Workshop on Swarm Robotics* (pp. 126–142). Berlin: Springer.
420. Winfield, A. F. T., Sa, J., Fernández-Gago, M.-C., Dixon, C., & Fisher, M. (2005). On formal specification of emergent behaviours in swarm robotic systems. *International Journal of Advanced Robotic Systems, 2*(4), 363–370. https://doi.org/10.5772/5769
421. Witten, T. A. Jr, & Sander, L. M. (1981). Diffusion-limited aggregation, a kinetic critical phenomenon. *Physical Review Letters, 47*(19), 1400–1403. https://doi.org/10.1103/PhysRevLett.47.1400
422. Wittlinger, M., Wehner, R., & Wolf, H. (2006). The ant odometer: Stepping on stilts and stumps. *Science, 312*(5782), 1965–1967.
423. Wolf, H. (2011). Odometry and insect navigation. *Journal of Experimental Biology, 214*(10), 1629–1641. ISSN 0022-0949. https://doi.org/10.1242/jeb.038570.

424. Wolpert, L. (1996). One hundred years of positional information. *Trends in Genetics, 12*(9), 359–364. ISSN 0168-9525. http://view.ncbi.nlm.nih.gov/pubmed/8855666
425. Yamaguchi, H., Arai, T., & Beni, G. (2001). A distributed control scheme for multiple robotic vehicles to make group formations. *Robotics and Autonomous systems, 36*(4), 125–147.
426. Yamins, D. (2005). Towards a theory of "local to global" in distributed multi-agent systems. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)* (pp. 183–190).
427. Yamins, D. (2007). *A Theory of Local-to-Global Algorithms for One-Dimensional Spatial Multi-Agent Systems*. PhD thesis, Harvard University, November 2007.
428. Yamins, D., & Nagpal, R. (2008). Automated global-to-local programming in 1-D spatial multi-agent systems. In L. Padgham, D. C. Parkes, J. P. Müller, & S. Parsons (Eds.), *Proceedings of 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Estoril, Portugal, May 2008.
429. Yang, C. N. (1952). The spontaneous magnetization of a two-dimensional Ising model. *Physical Review, 85*(5), 808–816. http://dx.doi.org/10.1103/PhysRev.85.808
430. Yasuda, T., & Ohkura, K. (2008). A reinforcement learning technique with an adaptive action generator for a multi-robot system. In *The Tenth International Conference on Simulation of Adaptive Behavior (SAB'08). Lecture notes in artificial intelligence*, July 2008 (Vol. 5040, pp. 250–259). Berlin: Springer.
431. Yates, C. A., Erban, R., Escudero, C., Couzin, I. D., Buhl, J., Kevrekidis, I. G., et al. (2009). Inherent noise can facilitate coherence in collective swarm motion. *Proceedings of the National Academy of Sciences of the United States of America, 106*(14), 5464–5469. https://doi.org/10.1073/pnas.0811195106. http://www.pnas.org/content/106/14/5464.abstract
432. Zahadat, P., Christensen, D. J., Katebi, S. D., & Støy, K. (2010). Sensor-coupled fractal gene regulatory networks for locomotion control of a modular snake robot. In *Proceedings of the 10th International Symposium on Distributed Autonomous Robotic Systems (DARS)* (pp. 517–530).
433. Zahadat, P., Hahshold, S., Thenius, R., Crailsheim, K., & Schmickl, T. (2015). From honeybees to robots and back: Division of labor based on partitioning social inhibition. *Bioinspiration & Biomimetics, 10*(6), 066005. https://doi.org/10.1088/1748-3190/10/6/066005
434. Zahadat, P., Hofstadler, D. N., & Schmickl, T. (2017). Vascular morphogenesis controller: A generative model for developing morphology of artificial structures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO'17, New York, NY, USA (pp. 163–170). New York: ACM. ISBN 978-1-4503-4920-8. http://doi.acm.org/10.1145/3071178.3071247
435. Zahadat, P., & Schmickl, T. (2016). Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition. *Adaptive Behavior, 24*(2), 87–101. https://doi.org/10.1177/1059712316633028
436. Zhou, G., He, T., Krishnamurthy, S., & Stankovic, J. A. (2004). Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services* (pp. 125–138). New York: ACM.

Index

H. Hamann, *Swarm Robotics: A Formal Approach*,
https://doi.org/10.1007/978-3-319-74528-2