The University Of Sheffield.

Automated Reverse Engineering of Agent Behaviors

Wei Li

Supervisors:

Dr Roderich Groß

Prof Stephen A. Billings

A Thesis Submitted for the Degree of
Doctor of Philosophy

30th September 2015

Artificial evolution
is the end of
engineering's
hegemony.

Kevin Kelly
Out of Control

Abstract

This thesis concerns the automated reverse engineering of agent behaviors. It proposes a metric-free coevolutionary approach—*Turing Learning*, which allows a machine to infer the behaviors of agents (simulated or physical ones), in a fully automated way.

Turing Learning consists of two populations. A population of models competitively coevolves with a population of classifiers. The classifiers observe the models and agents. The fitness of the classifiers depends solely on their ability to distinguish between them. The models, on the other hand, are evolved to mimic the behavior of the agents and mislead the judgment of the classifiers. The fitness of the models depends solely on their ability to 'trick' the classifiers into categorizing them as agents. Unlike other methods for system identification, *Turing Learning* does not require any predefined metrics to quantitatively measure the difference between the models and agents.

The merits of *Turing Learning* are demonstrated using three case studies. In the first case study, a machine automatically infers the behavioral rules of a group of homogeneous agents only through observation. A replica, which resembles the agents under investigation in terms of behavioral capabilities, is mixed into the group. The models are to be executed on the replica. The classifiers observe the motion of each individual in the swarm for a fixed time. Based on the individual's motion data, a classifier makes a judgment indicating whether the individual is believed to be an agent or the replica. The classifier gets a reward if and only if it makes the correct judgment. In the second and third case studies, *Turing Learning* is applied to infer deterministic and stochastic behaviors of a single agent through controlled interaction, respectively. In particular, the machine is able to modify the environmental stimuli (to which the agent responds) and thereby interact with the agent. This allows the machine to reveal all of the agent's entire behavioral repertoire and help reinforce the learning process. This interactive approach proves superior to learning only through observation.

Acknowledgments

Contents

1 Background and Related Work

This chapter presents the background and related work. In Section 1.1, we introduce the background of this thesis. This includes a brief introduction of the development of artificial intelligence (AI), how AI and robotics are combined, and the research that has been done in recent years in the areas of automation science. Section 1.2 reviews the development of evolutionary computation which is the main technique used in this thesis. This section includes an introduction of biological evolution, principles, strengths and caveats of evolutionary computing and its applications. Section 1.3 introduces how AI/robotics and animal behavior study benefit from each other, which includes how animal behavior can be used as inspiration for AI and robotics and the methods to investigate animal behavior using AI/robotics techniques.

1.1 Background

1.1.1 The Development of AI and Robotics

Intelligence is a natural part of life. Humans and other biological creatures exhibit many intelligent behaviors such as pattern recognition and decision making. However, intelligence is not a property that is limited to biological creatures. It should be equally applicable to computers or machines. The term AI emerged in a conference in 1956 at Dartmouth College, where several pioneers of this filed including Marvin Minsky, John McCarthy, etc., discussed the development of digital computer and the future of AI. The definition of AI is still a disputed topic. Some researchers argued that AI is to simulate the intelligent behaviors which are observed in humans and other biological creatures using computers or machines. That is, an intelligent machine should be able to exhibit behaviors similar to that of live creatures when encountering the same problems [1]. Others gave the following definition: "Artificial Intelligence is the study of mental faculties through the use of computational models" [2]. According to Fogel [3], an intelligent system should know how to make decision in order to fulfill a goal (e.g., solving a problem). In other words, instead of pre-programming the machine using human's knowledge, it should be able to learn and adapt. In [4], Minsky even argued, "Why can't we build, once and for all, machines that grow and improve themselves by learning from experience? Why can't we simply explain what we want, and then let our machines do experiments or read some books or go to school, the sorts of things that people do?" In 1950, Turing [5] proposed an imitation game which is nowadays known as *Turing test* to discuss a question: "Can machine think?". Although whether a machine could pass the *Turing test* or not is beyond the consideration at that time, it was accepted as a notion that a machine could learn and adapt to mimic human behavior. Many promising achievements have been made to enable machines to accomplish a variety of intelligent actions since then.

In the 1970s, the emergence of expert system—a computer program that mimics a human expert's decision-making capability [6], significantly promoted the development of AI. An expert system can solve complicated problems through reasoning about the knowledge (which is mainly represented as *if-else* rules) it has. One of the most representative examples is IBM's chess program (Deep Blue). It defeated the champion of the world chess (Gary Kasparov) in 1997 [7], which provides evidence that a computer program can even outperform a human expert in term of decision-making ability. An expert system consists of two components: knowledge base and inference engine. Knowledge base contains facts and rules that are known to the system. Inference engine uses the knowledge to make decision and derive new rules, which are then stored in the system to update the knowledge base. Expert systems have many commercial applications such as medical diagnosis [8], prediction [9] and monitoring [10], etc. The rules in an expect system can be expressed using Boolean logic or Fuzzy logic. In Boolean logic, every condition in the rules is either true or false. Fuzzy logic was introduced by Zadeh [11] to describe "degree of truth". For example, a cup with water is described as "full (1)" or "empty (0)" using Boolean logic; however, in Fuzzy logic, it can also be described using some fuzzy expressions such as "almost full", "half full", "near empty", etc. Fuzzy logic is used common in our daily life. An example rule in an expert system using Fuzzy logic is: *IF the temperature is cold, THEN turn the heater on.* In stock market an old saying is: "buy low, sell high". However, whether the stock value can be considered as low or high depends on the stock curves in a particular situation. Fuzzy systems have many commercial applications in such as air conditioner, digital camera and and hand writing recognition, etc. Another representation of AI is neural network, which mimics the processing ability of nervous systems of biological creatures (especially human brain). Through a combination of weights and excitation functions (e.g., sigmoid function), neural networks can accomplish many tasks observed in humans, such as pattern recognition and image processing.

Robotics is a field about making machines that can move in several ways to accomplish certain tasks. While AI and robotics are not essentially connected, they are often used together to make the robots "smarter". For example, a robot with AI can move autonomously and make decision while interacting with the environment it is operating on. Through combining AI and robotics, machines can be created to behave more like humans or animals. Instead of only executing fixed programs as in a car assembling line, robots can learn and adapt to the changing environment.

There are two common architectures adapted for the control of a robot: deliberative architecture and subsumption architecture [12]. In deliberative architecture, the robot operates on a top-down fashion and its action mainly depends on planning. A typical cycle is: $sensor \rightarrow plan \rightarrow act$. In the sensor stage, the robot gets the information from the world based on sensors such camera, infrared sensors. After pre-processing, this information would be passed to the central control architecture which integrates all the sensing information and reasons about it. Based on the knowledge the robot has to decide which action to take to fulfill a goal (e.g., maximize its reward). This architecture has led to many successful applications. The pioneer work is *Shakey the robot* which is capable of reasoning about its own actions [13]. In this work, the environment the robot is operating on is simplified and the experimental conditions are well controlled (e.g. uniform color and flat floor). More work has been done since then to enable robots to tackle complex and changing environments [14]. Another architecture adapted widely nowadays is subsumption architecture [12], in which the robot makes decision based on $sensor \rightarrow act$ without deliberate reasoning or planing. Instead of building a central reasoning system to integrate all the sensory information, the robot could process it in parallel by each layer. This could enhance the robustness of the robotic control system. Some famous robots using subsumption architecture are Allen [12], Herbert [15] and Genghis [16].

1.1.2 Introduction of Automation Science

With the development of AI and robotics, intelligent and automation systems were commonly used for assisting scientific research. Since the first clinical automated laboratory management system [17] was created in 1993, such systems are increasingly used in drug discovery, agriculture and energy labs, etc. Nowadays, it is the demand that machines could automate the whole process of scientific research. The question of whether it is possible to automatically conduct scientific research is very interesting in theory, and it also involves a lot of practical work which needs to be solved (e.g., communication and coordination). In order to speed up experimental process, researchers should take advantage of intelligent and automation systems to help, for example, collect and analyze thousands of data, because these things are very time-consuming and boring if only carried out manually. The ideal situation is to make a machine conduct scientific research automatically without or with little human intervention, and it can do experiments day and night in a constant manner without any tiredness and complain.

The field of automation science has been developed to a great extent because of the increasing demands of drug industry and relevant fields of biology and chemistry. High-throughput screening (HTS) systems [18] are one of the early efforts. HTS systems could do many things such as preparation, observation and analysis of assay, greatly enhancing the speed of data collection and analysis process in a short time. Recently, King, et al. have built a Robot Scientist—Adam, which can automatically generate functional genomics hypotheses about the yeast *Saccharomyces cerevisiae* and carry out experiments to test and refine the hypotheses based on the techniques developed in AI and robotics [19, 20]. This Robot Scientist is able to do plenty of experiments and observations a day. The experiments are performed automatically by machines, which makes it possible to test all aspects of experimental process. Adam could automatically conduct the cycles of scientific experiments: forming hypothesis, initializing the experiments, explaining the results and verifying the hypothesis, and then repeating the cycle. The functional genomics hypotheses are autonomously generated by intelligent software and the experiments are conducted coordinating different components in the automated system [19].

In system identification area, Gauld et al. have developed a digital automated identification system (DAISY) to identify biological species automatically with high accuracy using advanced image processing technique [21]. This technology has gone though great improvement in recent years, raising the possibility of automation, or at least semi-automation, in the process of routine taxonomic identification. In [22], MacLeod et al. reported that an imaging system that is originally designed for identifying marine zooplankton was used by the US government for monitoring horizon oil spill in the deep water. They argue that taxonomists and researchers in machine learning, pattern recognition as well as artificial intelligence should collaborate with each other in order to better identify and name biological species.

Drawing on approaches from various research areas especially AI and robotics, intelligent and automation systems are playing a vital role in scientific research, allowing researchers to conduct experiments more efficiently. It is argued that the revolution of automation science would emerge in a few decades [20].

1.2 Evolutionary Computation

Evolutionary computation is a technique which draws inspiration from biological evolution. It is a stochastic search method and uses trial-and-error to guide the search process. We use evolutionary computation technique to automate generation of models during the system identification process. Section 1.2.1 briefly introduces the biological evolution. Section 1.2.2 details the principles, strengths, weaknesses of evolutionary computation, including evolutionary algorithms and coevolutionary algorithms. Section 1.2.3 presents three main applications of evolutionary computation and the related work.

1.2.1 Biological Evolution

Biological evolution is about how living creatures evolve to adapt to their changing environment. According to Darwin's Theory of Evolution [23], species fight for survival. The species that can fit the environment survive, and others that can not would die. This phenomenon is regarded as *survival of the fittest* or *natural selection*. There are heritage (e.g., through sexual reproduction) and random mutation among species' genes. The genes that help the species survive would have a higher chance of be preserved and passed on to the next generation, while the genes that are harmful or not useful would be abandoned.

Table 1.1: The relationship between different species

	the influence of species A		
the influence of species B	+,+ (reciprocity)	+,0 (symbiosis)	+,- (predation)
	0,+ (symbiosis)	0,0 (neutral)	0,- (amensalism)
	-,+ (predation)	-,0 (amensalism)	-,-(antibiosis)

Natural selection tends to reserve and accumulate small beneficial genetic mutations. Suppose that some members in a species have evolved a functional organism that is very useful (e.g., a wing that can fly). This makes these members easier to find food or avoid threat from predators. Their offspring are more likely to inherit such advantageous function and this function would be passed to the next generation. The other members without the advantageous function are more likely to die out. Natural selection helps the species to compete and adapt better in the environment. At the same time, it also accelerates the extinction of the species that can not fit the environment. The dinosaur used to be a dominant species in the ancient world due to its big body and flying ability. This was a big advantage when the climate was mild. However, as the climate changed dramatically (e.g., extremely cold or hot), the big body was no longer an advantage as it needed to consume much more energy. This led to accelerate the extinction of dinosaurs.

Coevolution is special form of evolution, which involves in the simultaneous evolution of two or more species. A typical example of coevolution is fox and rabbit or parasite and host. In nature, the survival ability between species is coupled. That means the survival ability of one particular individual in a species depends not only on its chromosome, but also the interaction with the individuals from other species. Although the correlation between different species is complicated, for a specific species, there are three possibilities: beneficial, injured and neutral. Therefore, through permutation and combination, the relationship between two species can be summarized in Table 1.1. It includes 6 concrete relationship: reciprocity, neutral, symbiosis, amensalism, predation and antibiosis [24].

Where, symbol '+', '-' and '0' represent beneficial, injured and neutral. For example, "+, -" indicates A benefits and B gets injured in the relationship.

1.2.2 Introduction of Evolutionary Computation

1.2.2.1 Principles

Based on the principle of biological evolution, a bi-inspired algorithm — genetic algorithm (GA) was proposed by Halland in 1960s [25]. GA simulates the heritage, mating, and mutation of the natural evolution. In GA, the solution for a given problem is represented as chromosome, which contains several genes. Each gene could be a binary number, integer, or floating-point number. Heritage is also called reproduction in GA. Mating corresponds to crossover/mate, in which different individuals exchange genes. Mutation is normally realized by randomly changing a particular gene. For example, some gene in the chromosome may be randomly replaced by another gene. Mutation in GA serves the same function as it is in natural evolution. It creates the diversity and creativity among the population. GA is driven by a fitness function, which defines the goal or solution to be achieved. The evolutionary process is to optimize (e.g., maximize) the fitness of the individuals in the population. There are other types of evolutionary algorithms that based on the basic idea of natural evolution. Fogel, Owens and Walsh invented *evolutionary programming (EP)* [26]. Rechenberg and Schwefel introduced *evolution strategies (ES)* [27], which are mainly dealing with real-value continuous optimization problem. In the early 1990s, another new evolutionary algorithm called *genetic programming (GP)* was presented by Koza [28]. Fig. 1.1 shows a brief classification of evolutionary algorithms.

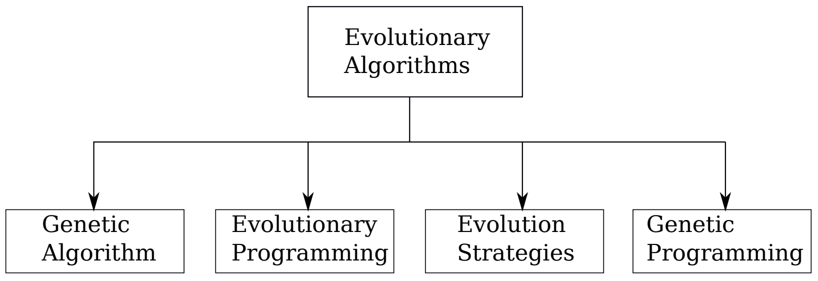


Figure 1.1: Classification of evolutionary algorithms

The implementation of evolutionary algorithms follow a general flow during the operation process. They can be divided into 5 steps: initialization, evaluation, mutation, selection and termination. At the beginning, a random population of individuals is initialized. These could be several random strings (chromosomes or individuals), each of which contains several genes (e.g., floating point number). These strings are encoded of the solutions to be achieved. Different chromosomes are then evaluated using the predefined fitness function. The fitness of individuals in the population is only decided by their own chromosomes. That means in different generations, the fitness of individuals who own the identical chromosome is constant. Selection happens after the evaluation of each individual, and the ones with higher fitness normally have a higher chance of being selected to the next generation and have offsprings. The offsprings would go through mutation, which helps to maintain diversity of the whole population. Fig. 1.2 show a diagram of how the evolutionary algorithms proceed.

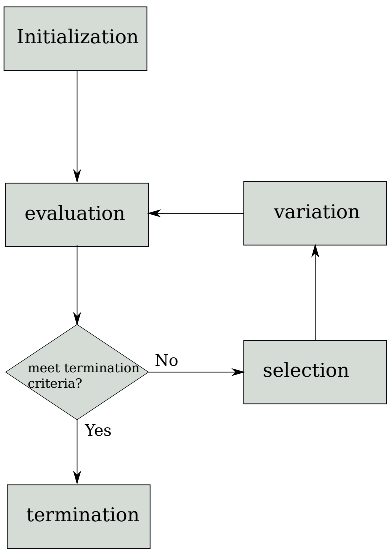


Figure 1.2: This diagram shows the flow of evolutionary algorithms.

As we mentioned before, the creatures in nature are not independent, and they have different kinds of relationship shown in Table 1.1. The evolution of individuals among different species are coupled. Based on the idea of biological coevolution, the coevolutionary algorithms, which coevolve simultaneously two or more populations, are widely used in various research areas [29]. Figure 1.3 shows a schematic diagram of coevolutionary algorithms between two different populations. In principle, coevolutionary algorithms can be considered of comprising several sub-algorithms, each of which could be an evolutionary algorithm. These sub-algorithms interact with each other in the fitness calculation process. In other words, the fitness of individuals in one population not only depends on its own chromosome, but also on the performance of other individuals from another population during the coevolutionary process.

The essential difference between evolutionary algorithms and coevolutionary algorithms

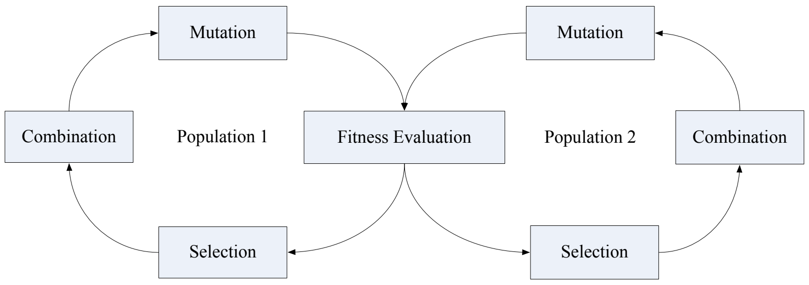


Figure 1.3: A schematic diagram of coevolutionary algorithms between two different populations.

is the way of fitness evaluation. While the fitness of an individual in evolutionary algorithm depends only one its chromosome, in coevolutionary algorithms the individual's fitness depends on its performance when evaluated by the individuals from the other populations. According to the way of evaluation, the coevolutionary algorithms are generally divided into two categories—competitive coevolutionary algorithm (Comp-CEA) and cooperative coevolutionary algorithm (Coop-CEA). Comp-CEA assesses each individual by its competitive performance with respect to its opponents, while Coop-CEA assesses each individual by its cooperative performance with respect to its co-operators. As discussed by Dawkins and Krebs [30], competitive coevolution can produce a phenomena of "arm races" by increasing the complexity of each population in the coevolution. The evolution of one population may drive another population to evolve new strategies, which makes both populations evolve a higher level of complex behavior. Generally, Coop-CEA is applied to the situation in which the problem can be divided into several sub-problems. In Coop-CEA, thare are several cooperative species evolving simultaneously, and each sub-species represent a part of the whole solution, which is the combination of the eventual solution in each sub-species according to a certain sequence. In the rest of the thesis, we only discuss Comp-CEAs, which will also be referred to as coevolutionary algorithms in general.

In coevolutionary algorithms, the fitness of individuals is called subjective fitness [31]. An individual's subjective fitness is based on the performance of its temporary opponents from the current generation or a combination of current and past generations. The fitness of individuals with the same chromosome in different generations may vary because of changing opponents. Conversely, the fitness in evolutionary algorithms is called absolute or objective fitness.

Comp-CEA can also be applied in single population or multiple populations. The single-population Comp-CEA is realized by competition between individuals within the same population. Suppose there are two populations in a coevolutionary algorithm. One is called "learner", and the other is called "evaluator". Let L represent a set of learners and E represent a set of evaluators. For a learner, its simple subjective fitness is the number of evaluators (in the current population) that this learner defeated [32]. It is described in equation (1.1) as follows:

$$\forall i \in L \Rightarrow CF_i = \sum_{j \in E, i\, defeats\, j} 1 \tag{1.1}$$

CF_i is the fitness of learner i.

Another fitness calculation approach is called competitive fitness sharing [29] as described in the following:

$$\forall j \in E \Rightarrow N_j = \sum_{k \in L, k\, defeats\, j} 1 \tag{1.2}$$

$$\forall i \in L \Rightarrow CF_i = \sum_{j \in E, i\, defeats\, j} \frac{1}{N_j} \tag{1.3}$$

N_j represents the number of learners that could defeat evaluator j.

In competitive fitness sharing [29], the learner that could defeat the more competitive evaluator get higher reward. For example, if a learner, i, in a population is the only individual to defeat an evaluator, j, this learner's accumulative fitness is added by 1, as N_j is equal to 1 in Equation (1.3). The aim of using competitive fitness sharing is to preserve/award the learner that possesses important genetic materials which are worth passing to the next generation.

There are many ways to choose the evaluators (temporary opponents). Random Paring [33] means finding a random temporary opponent for each leaner. In single elimination tournament [34], all the individuals randomly match, and the losers are taken out and winners are selected into next round of random match. Round Robin [33] means all the evaluators are the temporary opponents of each leaner. There are also other ways such as K-random opponent [34] and fitness sampling [29]. In fitness sampling, the selected temporary opponents should have relatively higher fitness value in the last generation. The evaluation time for random paring is the shortest, but the performance is the worst; the calculation time for round robin is the longest, but the performance is the best. In our thesis, we chose to use the simple subjective fitness calculation and Round Robin.

1.2.2.2 Strengths

The primary advantage of evolutionary computation is it is conceptually simple. Evolutionary computation draws inspiration from biological evolution, and it can be implemented using simple operators as mentioned before. Moreover, evolutionary computation is task-independent, which means it can be applied to virtually any task that can be formulated as a function optimization problem [35]. The advantage of function optimization as well as black-box optimization will be detailed in Section 1.2.3.1. Evolutionary computation (especially evolutionary algorithms) can be run in parallel (e.g., using GPU computing) to accelerate the evolutionary process. Each individual can be evaluated independently according to the predefined fitness function. Only the selection and recombination process need the serial computing. Evolutionary computation is robust to changes of circumstance. Once the circumstance has changed, the evolve solutions can be used as a start for further development without the need to restarting the whole process [36]. Apart from the general advantages of evolutionary computation, when compared with evolutionary algorithms, coevolutionary algorithms have the following two advantages:

- *Open-Ended Evolution* Coevolutionary algorithms can create an open-ended evolution for each population due to the complex interaction between the competing populations during the coevolutionary process. Such open-ended evolution could encourage the appearance of new building blocks, thus maintaining the diversity of populations. In Darwin's natural selection, this phenomenon is referred to as "arm race" [30], which leads each species to continuously improve.

- *No Absolute Fitness Needed* Coevolutionary algorithms can be applied to solve problems in which absolute fitness can not be effectively defined. For example, when evolving a chess program, it would be challenging to define a fitness to determine which program is better. An effective way of evaluating a chess program is making it play with other programs and then calculating its subjective fitness [32, 37].

1.2.2.3 Weaknesses

The main weakness of evolutionary computation is that it requires a sufficient number of generations to obtain good solutions. In simulation, this would not be a problem; however when the evaluation needs to be conducted on the real system, it would take a long time and may also cost a lot. If the fitness function can not be defined properly, the obtained solutions may be biased. Although coevolutionary algorithms have some advantages over evolutionary algorithms, there are three main pathologies:

- *Red Queen Effect* During the coevolutionary process, two populations keep competing with each other. For a particular population, when *Red Queen Effect* happens, the variation tendency of subjective fitness and objective (absolute) fitness is opposite [38]. For example, the increase of its subjective fitness may correspond to the decrease of its objective fitness. In other words, the objective fitness does increase, but the landscape of subjective fitness does not reflect such situation.

- *Cycling* In coevolutionary algorithms, the aim of each individual is to defeat its temporary opponents. The optimal solution which is obtained in the previous generations would probably be lost. After some generations, the optimal solution may be found but lost again. The coevolution is trapped into this endless cycling, failing to find the optimal solution [31]. The general way of overcoming the problem of "cycling" is "hall of fame"[29]. "hall of fame" obtains the excellent individuals from the previous generations, and these obtained individuals may be selected to be temporary components to evaluate the individuals from the competing population in the current generation.

- *Disagreement* During the coevolutionary process, when one species is entirely better than the other, disengagement will occur. In this case, the selection criteria won't make any sense and selection gradient will disappear, since the subjective fitness of each population is constant. The diversity of populations will converge into zero, and it is impossible to form "arm race" [30]. The common solution for solving the disengagement problem is "resource sharing" and "reducing virulence"[31]. "Resource sharing" keeps the diversity of populations, and "reducing virulence" selects the evaluators to keep the gradient of learners' fitness.

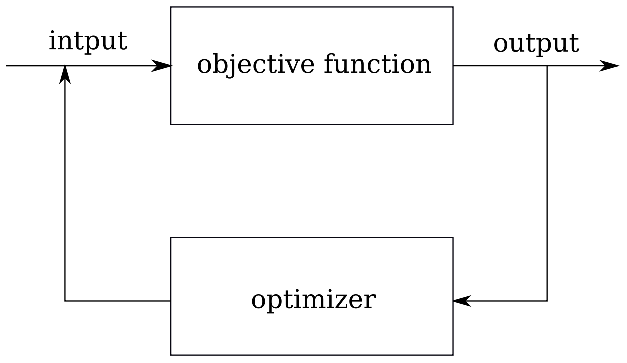


Figure 1.4: This diagram shows the process of black box optimization. The task is to find a candidate solution (input) that can optimize (e.g. maximize or minimize) the objective function.

1.2.3 Applications of Evolutionary Computation

Evolutionary algorithms are widely used for solving various engineering tasks ranging from optimization, control, pattern recognition, robotics and system identification/modelling. In the following sections, we will focus on three main fields in which evolutionary computation technique plays a role.

1.2.3.1 Black Box Optimization

One of the promising application of evolutionary computation is black box optimization. Black box optimization refers to such problems that the optimization algorithm aims to optimize an objective function without assuming the hidden structure of that function (e.g., linear or differentiable). In particular, the aim is to found a set of inputs that can maximize or minimize the output of the objective function. For example, in a traveling salesman problem, in which the task is to find the best combination of route of cities (input) that can minimize the length of tour of visiting all cities. Fig. 1.4 shows a diagram of the black box optimization.

Due to the complexity of the real problem in nature, the stochastic search algorithm such as evolutionary algorithms provide us an efficient and relatively 'perfect' solutions. Although evolutionary algorithms could not guarantee the best solution would be found every time, they are still superior to many traditional search algorithms such as the greedy local search algorithm [39]. There are many real-world examples that using evolutionary computation techniques to solve the optimization problem. In the area of nanophotonic light trapping, an urgent need is the development of low cost thin film solar photovoltaic technologies. A traditional way is fixing the structure according to the physical intuition and trying to find the optimal parameters. In [40], a highly efficient light-trapping structure was designed using a genetic algorithm. It was shown that this new structure can increase the trapping efficiency three times comparing with the classic limit. The high efficiency achieved by the new design is far beyond the reach of traditional design. Another successful example is using evolutionary algorithms to design antennas for NASA's Space Technology spacecraft [36], and one of the antennas was used in the mission. The antennas is a critical device for the spacecraft to communicate with the ground, as faulty communication may cause a lot of data lost or the crash of the spacecraft. The antennas designed using evolutionary algorithms are significantly better than those designed by human experts.

In the following, we list several cases (but not all) that evolutionary computation could be applied to solve the 'tough' optimization problem.

- *High-dimension* As the dimension, n, of the objective function increases, the search space increases exponentially. This is called "curve of dimension" by Bellman [41]. For example, if we have to optimize a function that has 30 dimensions, and each dimension only has 20 parameters to be selected. For a grid search in which all the possible solution is evaluated, it will take 20^{30} evaluations. Suppose that each evaluation takes $1\mu s$, it would more than the $3 \cdot 10^{31}$ years. However, if using evolutionary computation, it probably takes hours to find the optimal solution.

- *Multi-Model* Multi-model means a system (function) has more than one optimums (e.g., Rastrigin Function). The one/s with the best fitness value is/are considered as global optimum/s, and the other are considered as local optimums. These local optimums around the global optimum/s are very misleading for the gradient-based search algorithms (e.g., *hill climbing* algorithm), as the solutions may easily get trapped at the local optimums. Evolutionary algorithms are shown to be very efficient to find the global optimum/s [44].

- *Non-separable and non-differential* A function, $g(x_1, x_2, \cdots, x_n)$ is non-separable, if it can not be expressed as: $g(x_1, x_2, \cdots, x_n) = g(x_1)g(x_2)\cdots g(x_n)$. For the separable function, it would be much easier to optimize, as we can treat each variable separately. However, for non-separable system, the variables are normally coupled. Also, the function may not be differential, which makes many mathematical optimization algorithms (e.g., quasi-Newton BFGS algorithm [42] or conjugate gradient algorithm [43]) infeasible.

- *Multi-objective* When encountering real-world problem, we usually need to deal with multiple objectives, which need to be optimized simultaneously. For example, when designing a car, the engineers need to consider the shape, performance of the engines, and more importantly cost. As some of the objectives (such as performance and cost) are conflicting, the designer needs to find a Pareto optimal solution [45], in which none of the value of the objective functions can not be increased without considering decreasing the value of other objectives. Evolutionary multi-objective optimization is an efficient technique for generating such Pareto optimal solutions for multi-objective optimization problems. For a review, see [45].

1.2.3.2 Evolutionary Robotics

Another application of evolutionary computation is evolutionary robotics, in which the controller and/or morphology of the robots are automatically generated. The user considers the robot as a whole and only needs to specify the criterion which is represented by a fitness function. The fitness function could be single-objective or multi-objective. The aim is to optimize (minimize or maximize) the fitness using evolutionary algorithms. The normal procedure of evolutionary robotics is as follows. First, an initial population of random controllers are generated. Each controller are represented by a chromosome. If the controller is a neural network, the chromosome is a vector of real numbers. Each controller was imported into the robot(s), and the robot's performance when performing various tasks is measured and evaluated using the pre-defined fitness function. After some genetic operators (e.g. crossover, mutation), the 'fitter' robot controller has a higher chance of being selected and generating offspring in the next generation. This process iterates until a good solution is found.

In contract to evolutionary robotics, another method of designing robot controller and/or morphology is behavior-based approach, where the designer divides the whole system into several simple parts intuitively. These separate parts are then integrated all in once through a coordination mechanism—competitive or cooperative coordination [12]. In competitive coordination, only one part has an effect on the output of the robot, while in cooperative coordination, several parts contribute to the output of the robot with different weights. The behavior-based method was shown to be very robust in many tasks such as gait control in locomotion and object transport. The challenge of designing robot controller and/or morphology using behavior-based method is it requires a lot of experience from the designer. Moreover, when the system is highly coupled, separating the whole design into different parts may not be a good strategy. However, evolutionary robotics provides the designer an alternative way of controlling the robot as a whole, rather than focuses on the details of each separate component. The synthesized control system is a result of self-organized process.

There are many control structure that can be adapted in evolutionary robotics. One popular structure is artificial neural network due to its simple representation and strong power of control. The tree-based structure like LISP is also very robust, and it is widely used in evolutionary programming [46]. The *building blocks* method was proposed by Brooks [47]. However, these blocks are described using high-level languages, which are are suitable for evolution in the low level. Comparing with other control structures, artificial neural network have many advantages [48, 49]. According to the types of behavior (e.g., simple perception or non-linear dynamic), we can choose different neural networks (forward neural networks and recurrent neural networks, etc.). Recent development reveals that we can even evolve the topology and weights of the neural networks [50].

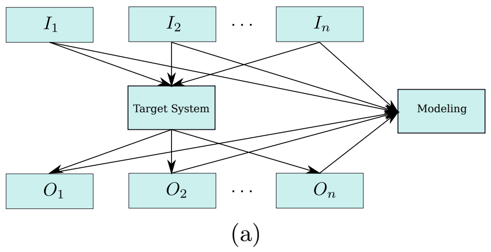
There are two main research aims in evolutionary robotics: 1) engineering—developing the control strategy for robots; 2) biology—to understand the biological systems using simulated evolution. In engineering, a range of work has been presented, ranging from simple behaviors, such as phototaxis behavior, self-charging behavior to complex behaviors such as navigation and locomotion of robot with multiple degrees of freedom. In biology, evolutionary robotics is used as a tool to understand the general principles of evolution. It provides much more efficient and faster way to validate or even create hypothesis based on evolution in simulation or real robots, compared with the slow evolutionary process in nature. AVIDA [51] and AEvol [52] are two computer software systems that are used for studying the evolution of bacterial. In hypothesis validation, evolutionary robotics was used as a tool to investigate some key issues in biology. For example, whether altruistic plays an important role in cooperation [53, 54], the conditions of emergence of communication during the evolution [55], and how morphology and control are coupled [56], coevolution of predator and prey [57, 58].

Using evolutionary algorithms to generate a desirable behavior/result requires a relatively large population size and certain number of generations. Therefore, performing evolution on the real robot would take plenty of time. The initial experiments may also cause damage to the robot itself or the environment the robot is operating on. Therefore, a lot of experiments/evaluations are performed in simulation. As the simulator can not match the reality, the controller generated in simulation may not work well in reality, causing *reality* gap. Some work are done to reduce the *reality gap*. In [59, 60], the author uses the transferability approach to increase the quality of controller generated in simulation through reducing the gap between simulation and reality. In [61], the evolution is even performed directly on the real robot to evolve the controller to perform navigation task. However, the battery is still a problem as it takes two weeks to evaluate 100 generations. The addressed this issue through connecting a wire between the robot and the power station.

Recently, a distributed online onboard evolutionary method (artificial embodied evolution) has received much attention [62, 63, 64]. In artificial embodied evolution, the population is distributed among different robots, and the gene exchange is done through *mating*. Each robot only exchange its genes with its nearby neighbors. There is not central control over the group of robots. This approach is particularly suitable for the situation where the environment that the robots are operating on is not predictable or changing after deploying the robots. The robots need to evolve to adapt to the changing environment, while satisfying certain basic requirements inserted by the designers. A more challenging research area could be evolving both the morphology and controller of the robots in a distributed manner—*evolution of things* [65]. This forms an open-end evolution among different artifacts, which is a process how living creatures evolve in real world. The fast development of 3D printing technique makes this method appealing [66].

1.2.3.3 System Identification

System identification which is about building the model of a hidden system through conducting a set of experiments is widely used in both academic and industrial areas [67]. The experiments are conducted by a series of inputs into the system and the outputs corresponding to the inputs are collected. The process of system identification is to find a model that fits the inputs and output. It is composed of observed data, model structure, a criterion to evaluate different models according to the observed data, validation of the obtained model based on different data set, and revision of model if necessary.



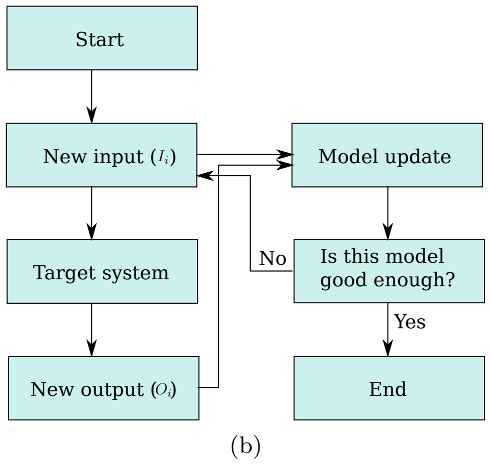


Figure 1.5: The diagram showing the two approaches ((a): offline; (b): online) for system identification. (I_i, O_i), where $i \in [1, 2, \cdots, n]$, represents a pair of input data and output data.

There are two system identification approaches: the offline approach and online approach. In the offline approach, the observation data is collected first, and the aim of modeling is to generate a model that fits the observed data. This method is often used when the data is easy to collect or the parameters and operating environment of the system do not change too much in a short time. However, in some cases, the parameters of the system are always changing due to different operating environment. That means the obtained model based on the previous observation data can not be applied to the new situation any more. In this case, the model should be updated using the new observed experimental data and online system identification method. The two approaches of system identification are shown in Fig. 1.5. Note that for some systems, there are only outputs.

System identification can be divided into two main procedures: modeling and estimation. Modeling defines the order or general structure of the hidden system [68]. Estimation identifies the parameters associated with the given structure. There are many estimation algorithms to determine the parameters for a given structure. Typical estimation algorithms are recursive prediction error methods which are based on gradient search. Gradient search method such as greedy algorithm can easily get trapped in local optima, especially when there are numerous local optimal points near the global optima. An alternative search method is using evolutionary algorithms. As evolutionary algorithms use population-based search method, this makes it more likely to get rid of local optima.

There are also some system identification methods that combine the modeling and estimation process, e.g., NARMAX method [69]. Neural network is a good representation of the system under investigation, as its topology and weights can be optimized simultaneously. The disadvantage of neural network is it is hard to interpret the obtained model when there are multiple layers, which makes it much hard to analyze the target system. Genetic programming provides an alternative way for finding the structure and parameters of the system. The model represented by genetic programming is described in a tree-based structure. As the structure in genetic programming is evolving, the obtained model may have different structure in different runs [70]. Bloating is a problem in genetic programming, where the growth of the tree increases the complexity of the model structure.

Coevolutionary algorithms provide an effective way for system identification [71], [72, 73, 74, 75, 76, 77]. A range of work have been performed on simulated agents. In [72], Bongard and Lipson proposed the *estimation-exploration algorithm*, a nonlinear system identification method to coevolve inputs and models in a way that minimizes the number of inputs to be tested on the system. In each generation, the input that leads to the highest disagreement between the models' predicted output in simulation was carried out on the real system. The quality of the models was evaluated through quantitatively comparing the output of the real system and the models' prediction. The method was applied to evolve morphological parameters of a simulated quadrupedal robot after it undergoes 'physical' damage. In a later work [73], they reported that "in many cases the simulated robot would exhibit wildly different behaviors even when it very closely approximated the damaged 'physical' robot. This result is not surprising due to the fact that the robot is a highly coupled, non-linear system: thus similar initial conditions [...] are expected to rapidly diverge in behavior over time". They addressed this problem by using a more refined comparison metric reported in [73]. In [74], an algorithm which is also based on coevolution of models and inputs was presented to model the simulated quadrotor helicopter and improve the control quality. The inputs were selected based on multiobjective performances (e.g., disagreement ability of models as in [72] and control quality of a given task). Models were then refined through comparing their prediction to each selected test trajectory. In [78], the damage detection process is conducted by coevolutionary algorithm to extract the maximum information from the system. In [79], coevolutionary algorithm is applied to estimate chaotic time series, in which the test data that can extract information from the chaotic system co-evolves with the models. In these works, predefined metrics are critical for evaluating the performance of models.

Many studies also investigated the implementation of evolution directly in physical environments, on either a single robot [80, 81, 82] or multiple robots [83]. In [80], a four-legged robot was built to study how it can infer its own morphology through a process of continuous self-modeling. The robot ran a coevolutionary algorithm on-board. One population evolved models for the robot's morphology, while the other evolved actions (inputs) to be conducted on the robot for gauging the quality of these models through comparing sensor data collected. In [83], a distributed coevolutionary approach was presented to coevolve on-board simulators and controllers of a swarm of ten robots to perform foraging behavior. Each robot has its own simulator which models the environment. The evolution of each robot's simulator was driven by comparing the real-world foraging efficiency (a pre-defined fitness metric) of its nearby neighbors each executing the best controller generated by their own simulators. Each robot has a population of controllers, which evolved according to the robot's on-board simulator. The best controller was chosen for performing real-world foraging. This physical/embodied evolution helps reduce the *reality gap* between the simulated and physical environments [84]. In all of the above approaches, the model optimization is based on pre-defined metrics (explicit or implicit), which are task dependent.

1.3 Combining AI/Robotics and Animal Behavior

The variety of animal behaviors in nature is immense, ranging from simple perception to complicated behaviors such as navigation and communication. The scientific study of animal behavior is pursued not only because it is a subject of interest in itself, but also because the knowledge gained from it has several practical applications. In AI/Robotics, there is a large interest of studying animal behavior, as the model/knowledge learned can be used to build a more intelligent machine. At the same time, building machine that mimics the animal helps us better understand its behavior. In this section, we review how AI/robotics and animal behavior study benefit each other. In Section 1.3.1, we introduce some interesting animal behaviors observed in nature. In Section 1.3.2, we detail how the animal behavior observed in nature can be used as inspiration for solving engineering tasks, especially in the area of AI/robotics. In Section 1.3.3, we show how AI/Robotics can contribute to the study of animal behavior, which is the theme of this thesis.

1.3.1 Animal Behavior in Nature

Comparing with the behaviors exhibited by complex animals such as mammals, insect behavior arises a large interest both by biologists and roboticists. One reason is due to the simple neural system of the insects, which provides a good inspiration for engineering purpose. This simplicity also makes it easy to replicate the intelligence exhibited by the insects.

A basic insect behavior is taxis, which is its intrinsic behavioral response to a specific stimulus. Taxis is divided into different type according to the stimulus which elicits the response. These behaviors include phototaxis (light), chemotaxis (chemicals), thermotaxis (temperature), etc. For example, a lobster follows the salt-water plume—a kind of chemical signal to find its source. Another interesting taxis behavior of crickets is that the female crickets perform complex auditory orientation behavior towards the male crickets. Researchers have found this complex sound localization behavior emerge from simple reactive steering responses to specific sound pulses generated by male crickets [85]. Apart from taxis behaviors, some insects use the stimuli as cues for navigation or migration. For instance, the bee uses its vision system to navigate in the air and avoid obstacles. Another interesting behavior found in insects is the ball movement of dung beetles [86]. Once the dung beetles form the pieces of dung into a ball, they always roll the dung-ball in a straight line using various stimuli (e.g., the moon, sun and polarised light [87, 88]) as visual cues to transport the food source. This behavior ensures that they keep away from the competitors as far as possible.

The stimulus-response behaviors in insects mentioned above are investigated by biologists for centuries. When investigating such behaviors, biologists need to learn how to interact with the animal in a meaningful way to extract all the behavioral repterio of the insect under investigation. However, whether this interacting ability could be exhibited by an intelligent machine is still an uncertain problem, which is addressed in this thesis.

Apart from single-animal behavior, swarm behaviors, which are emergent (collective) behaviors that arise from the interactions of a number of animals (especially social insects) in a group, have also been widely observed in nature. The individual behaviors in a swarm tend to be relatively simple [89]. The global behavior that is exhibited in a swarm is a result of self-organized process. Researchers found that individuals do not need the representation or complex knowledge to build a map of what the global behavior should be [90]. There are no leaders in the colony. From the point of control, swarm behavior is a distributed control system which does not rely on central coordination.

Many swarm behaviors are observed in nature. For example, the flocking behavior of birds are of particular interest to humans. This is not only because of their beautiful shape formed in a group, but also how the birds coordinate with each other to maintain that shape. A simple mathematical model was proposed by [91] to describe the individual behavior of each bird in a flocking. The three rules are: attraction, repulsion and alignment. In the attraction rule, the birds will be attracted by their neighbors, and this would result in each bird moving towards to the ‘center’ of their neighbors. Repulsion means the bird needs to avoid colliding with each other. Alignment assumes that each bird moves in the same direction with its neighbors. Although there is no prove that flocking birds follow exactly these three rules, it is attractive that the boids (in simulation) following such simple rules can mimic the real flocking behavior very closely. There are many some other swarm behaviors which are also studied extensively such as the aggregation of cockroaches [92], foraging in ants [93], flashing synchronization in fireflies [94], mound building in termites [95].

1.3.2 Swarm Optimization and Swarm Robotics

In the previous sections, we introduce some interesting animal behaviors observed in nature. Many algorithms are inspired from observation of animal behaviors. In this section, we will review two main optimization algorithms inspired from swarm behaviors—ant colony optimization algorithm [96] and particle swarm optimization algorithm [97]. Swarm robotics is another application area which uses the swarm behaviors of social animals as an inspiration to solve complex tasks using multiple robots.

1.3.2.1 Swarm Optimization

Ant Colony Optimization

Ant Colony Optimization (ACO) is an optimization technique that gets inspiration from foraging behavior of ants. When ants go out to search for food, they will leave pheromone in the path. Ants will be attracted by the pheromone, the strength of which represents the quality of the food source. Researchers have found that this indirection communication, which is know as *stigmergy* [98], leads them to found the shortest path along their nest and location of food source. The initial application of ACO is to find the optimal path in the combinational (discrete) problem. Note that nowadays the application of ACO algorithms ranges from network optimization (e.g., routing and load balance [99]) to continuous optimization [100].

The principle of ACO algorithms can be divided into the following two steps:

- Use the pheromone model to generate condidate solutions. From the aspect of mathematics, the pheromone model is a parameterized probability distribution in the search space.
- The candidate solutions are used as a bias for the future sampling to get better solutions.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is another an optimization algorithm which gets inspiration from flocking of birds or schooling of fish. It was first proposed by Kennedy and Eberhart [97]. The initial application is to optimize the weights of neural networks—a continuous optimization problem. It is also widely used in discrete optimization.

The basic component in PSO is called *particle*. A PSO algorithm consists of a finite set of particles. The movement of each particle is updated using *velocity*. The velocity of each particle in each time step is updated based on its current velocity, the deviation between the best position (it has found so far) and its current position, and deviation between the best position by its neighbors and its current position. This will result in the particles moving towards the high-quality solutions after certain iterations. The update of each particle can be written using two equations as follows:

$$\overrightarrow{v}_{i+1} = \overrightarrow{v_i} + c_1\overrightarrow{R_1} \otimes (\overrightarrow{p_i} - \overrightarrow{x_i}) + c_2\overrightarrow{R_2} \otimes (\overrightarrow{p_g} - \overrightarrow{x_i}) \tag{1.4}$$

$$\overrightarrow{x}_{i+1} = \overrightarrow{x_i} + \overrightarrow{v_i} \tag{1.5}$$

where $\overrightarrow{R_1}$ and $\overrightarrow{R_2}$ are independent random number generators that return a vector of random values in range $[0, 1]$. c_1 and c_2 are referred to as acceleration coefficients. The first item in Eq. (1.4) keeps the particle moving in the previous direction; the second item makes the particle move towards the best position of its own; the third position forces the particle move towards to the best position that its neighbors have found. Eq. (1.5) updates the particle's position.

1.3.2.2 Swarm Robotics

In the previous sections, we talk about how swarm behaviors can be used as inspiration for algorithms design. In this section, we introduce how to use swarm intelligence techniques to multiple robots research, which is referred to as swarm robotics. Many social insects' (e.g, ants, termites, wraps and bees) behavior can be used as inspirations in swarm robotics.

Swarm robotics investigates how multiple robots each with limited ability communicate, coordinate and self-organize to accomplish complex tasks. To finish the same task, using a single expensive robot with complex control structure may be feasible but it may have low efficiency and prone to failure. The advantages of swarm robotics are as follows:

- *Robustness* A swarm system consisting of multiple robots is robust in term of failure of certain robots disturbance of the environment. This robustness can be explained in the following reasons: 1) if some robots failed, the other robots would replace the functions of the failed robots; 2) the control is distributed; 3) as the individual is simple, it is less likely to be damaged; 4) the perception from multiple robots would increase the system's robustness. Note that in some swarm systems, there may be exceptions. That is, some individual failure would influence the whole self-organized process [101].

- *Scalability* In swarm robotics, the number of robots do not have significant difference on the global performance/behavior in the system. That is, increasing/decreasing certain number of robots, the system is still under control and the coordination is maintained. When investigating a swarm robotic system, scalability study is usually considered [102, 103].

- *Flexibility* The swarm could easily adapt to the changing tasks and generate relevant solutions [104]. The role of each robot in the swarm could be changed depending the need for the task.

In order to cooperate, the robots need to interact with each other and environment. There are three kinds of interaction in swarm robotic systems.

- *interaction via environment* In this interaction method, the robots communicate with each through changing the environment. There is no explicit communication between each robot (i.e., they do not exchange message). In nature, this method is observed in social ants. The pheromone they leave when foraging is an environmental stimulus for locating the food source.

- *interaction via perception* In this method, the robots can perceive each other in a limited range. This perception is local and there is no explicit communication between each robot. This requires the robots can distinguish between robots and objects in the environment. In nature, when the ants need to collectively pull the food to the nest, they need to perceive each other (to avoid collision) and the food (object).

- *interaction via explicit communication* In this method, a network is required to communicate with a swarm of robots in real time. This could be done by broadcast (e.g., WiFi [105]) or a distributed sensing network [106]. How to build a reliable network when the number of robots is significantly large is still a hot topic widely discussed. When the number of robots increases, the load of communication increases exponentially. A possible solution is combining the advantage of network communication and local communication using the robots' perception.

A range of tasks have been demonstrated in swarm robotics. The tasks range from aggregation [107, 108, 109, 92], dispersion [110, 111], pattern formation [112, 113], collective movement [114] to cooperative transport [115, 116, 117, 102], etc. Aggregation can be considered as the fundamental behavior of other more complex tasks. In [92], a group of robots mimic the cockroaches' aggregation behaviors, in which the robots gather into or leave the nest with a probability proportional to the size of the nest. The advantage of using stochastic algorithm is that they do not need to form a connected network in the initial configuration. In [108], the robots each with a binary sensor were reported to aggregate into a single cluster, validated using 40 e-puck robots. The robots do not need to perform algorithmic computation. This work was scaled well using 1000 robots in simulation. In [118], Werfel et al. designed a group of termite-inspired robots that work collectively to build several structures. The robots communicate with each other using *stigmergy*. In [114], a group of nine Kobot robots were reported to mimic the flocking behavior of birds. These robots followed some simple rules similar to those proposed by Reynolds [91]. In cooperative transport, Chen et al. [102] proposed a strategy in which the robots only push the object when the robots' vision of the goal is occluded by the object. This strategy was proved to push any convex object in a plenary environment.

1.3.3 Contribution of AI/Robotics to Ethology

In the previous sections, we reviewed how animal behavior study can be used as inspiration for designing algorithms and robotic systems. However, robotics can also benefit animal behavior study. In this section, we review two approaches in AI/robotics to contribute to the study of animal behavior: *learning from synthesis* and *robot-animal interaction*.

1.3.3.1 Learning from Synthesis

Ethologists have studied animal behavior over a century. However, even some basic behaviors such as taxis (which is widely observed in animals such as insect larvae and worms [119, 120]) is still not completely understood [121]. There are some basic steps that ethologists follow in the study of animal behavior [122]. The first step is observation, and after that they formulate some scientific questions on the observed behavior, and generate hypothesis to answer these questions. In order to verify the hypothesis, they actively conduct related experiments in the real animals and collect certain amount of data. After analyzing the data, the conclusion will be made to support or reject their hypothesis.

Robotics or artificial life can be used as an alternative methodology to investigate and understand animal behavior. Robots can be used as physical models of animal behaviors for testing hypotheses [123, 124]. For example, taxis behavior has often been implemented on mobile robotic systems. Webb [125] used a robot to model the phonotaxis behavior of crickets [126]. The robot can locate the position of a sound source and move towards it under different conditions. There was a good agreement between data collected from experiments on the robot and the animal. Another taxis behavior—chemotaxis in which animals follow a specific chemical trial has been used as a model for robots to find odour source based on artificial neural networks [127] and even Braitenberg vehicles [128]. Robots can be used as a validation for the models obtained from biologists and allow them to better understand the animal behavior from a synthetic point of view. Besides, roboticists can generate new hypotheses and test them using (simulated or real) robots.

In social behaviors study, Balch et al. [129] built executable models of the behaviors and ants and monkeys, which can be directly executed by multi-robot systems. The aim is to show how research into multi-robot systems can contribute to the study of collective animal behaviors. Besides robotics, there are some researchers arguing that artificial intelligence can also make significant contributions to biological study. In [130], Chappell et al. argue that there are many ways in which biologists interested in natural intelligence can learn from AI, and they outline the many specific kinds of contributions that AI can make to biological study. They also give some suggestions on how AI researchers collaborate with biologists to combine the advantage of each other and solve some cutting-edge problems in animal behavior.

As opposed to the works mentioned above, the method proposed in this thesis aims to synthesize models of animal (agent) behaviors automatically, rather than manually. This could help to spare scientists from having to perform numerous laborious experiments, allowing them instead to focus on using the generated models to produce new hypotheses and conduct further experiments.

1.3.3.2 Robot–Animal Interaction

Besides pure robot-based or AI research, researchers also use robots to interact with real animals. They build and programme robots (i.e., replicas) that can be inserted into the group of social animals [131, 132, 133, 134, 135]. Robots can be created and systematically controlled in such a way that they are accepted as con- or hetero-specifics by the animals in the group [136]. In this case, one "animal" in a group is completely controlled and they can observe the behaviors of the mixed society [137]. The behavior of the inserted robot can be controlled and the model can also be embedded into the robot for verification [138]. The behavior of robots can be programmed in such a way that its behavior is not influenced by the other real animals in the group, and they can be used as demonstrators or leaders in the experiments. Further more, it is easier to verify a hypothesis through controlled interaction in social behaviors.

For example, in [131], a replica fish which resembled the appearance (i.e., visual morphology) of sticklebacks was created to investigate two types of interaction: recruitment and leadership. In [133], autonomous robots which executed the derived model were mixed into a group of cockroaches to modulate their decision-making of selecting shelter in the aggregation behavior. The robots behaved in a similar way to the cockroaches. Although the robots' appearance was different to that of the cockroaches, the robots released a specific odor that the cockroaches could detect and regard the robots as conspecifics. In [139, 135], Vaughan et al. have built a mobile robot that can interact with the ducks in a circular arena and drive them to the safe place. Halloy et al. In [140], Gribovskiy et al. designed a robot which is capable of interacting with chicks to study how the behavior of chicks can be influenced by the others in a group. Although robots which are well designed can be mixed in social animals, building such kind of robot is a time-consuming process. It is also expensive to some extent and requires the collaboration of researchers from different disciplines. In [141], a robot-fish that can interact intelligently with live zebrafish to study their preference and locomotion behavior was designed .

In these works, the models were manually derived and the robots were only used for model validation. We believe that this robot-animal interaction framework could be enhanced through *Turing learning*, which autonomously infers the collective behavior.

Bibliography

[1] H. Schildt, *Artificial intelligence using C.* New York, NY, USA: McGraw-Hill, 1987.

[2] E. Charniak, *Introduction to artificial intelligence.* Reading, MA, USA: Addison-Wesley, 1985.

[3] D. B. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence.* Street Hoboken, NJ, USA: Wiley-IEEE Press, 1995.

[4] M. L. Minsky, "Logical versus analogical or symbolic versus connectionist or neat versus scruffy," *AI magazine*, vol. 12, no. 2, pp. 34–51, 1991.

[5] A. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[6] P. Jackson, *Introduction to expert system.* Boston, MA, USA: Addison-Wesley, 1998.

[7] M. Newborn, *Kasparov versus Deep Blue: Computer Chess Comes of Age.* New York, NY, USA: Springer-Verlag, 1997.

[8] J. H. Connell, *Minimalist Mobile Robotics.* Burlington, MA, USA: Morgan Kaufmann, 1990.

[9] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, "Dendral: A case study of the first expert system for scientific hypothesis formation," *Artificial Intelligence*, vol. 61, no. 2, pp. 209 – 261, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/000437029390068M

[10] P. Salvaneschi, M. Cedei, and M. Lazzari, "Applying ai to structural safety monitoring and evaluation," *IEEE Expert*, vol. 11, no. 4, pp. 24–34, Aug 1996.

[11] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[12] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14–23, Mar 1986.

[13] N. J. Nilsson, "Shakey the robot," SRI International Technical Note, Tech. Rep., 1984.

[14] V. Dimitrov, M. DeDonato, A. Panzica, S. Zutshi, M. Wills, and T. Padir, "Hierarchical navigation architecture and robotic arm controller for a sample return rover," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, Oct 2013, pp. 4476–4481.

[15] B. G. Buchanan and E. H. Shortliffe, *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley Series in Artificial Intelligence).* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984.

[16] R. A. Brooks, "A robot that walks; emergent behaviors from a carefully evolved network," Cambridge, MA, USA, Tech. Rep., 1989.

[17] M. Steinlechner and W. Parson, "Automation and high through-put for a dna database laboratory: development of a laboratory information management system," *Croatian medical journal*, vol. 42, no. 3, pp. 252–255, 2001.

[18] A. Persidis, "High-throughput screening," *Nature biotechnology*, vol. 16, no. 5, pp. 488–493, 1998.

[19] K. E. Whelan and R. D. King, "Intelligent software for laboratory automation," *Trends in Biotechnology*, vol. 22, no. 9, pp. 440–445, 2004.

[20] R. King, J. Rowland, S. G. Oliver, and M. Young, "The automation of science," *Science*, vol. 324, no. 5923, pp. 85–89, 2009. [Online]. Available: http://www.sciencemag.org/content/324/5923/85.abstract

[21] N. Gauld and Gaston., "Driving miss daisy: The performance of an automated insect idenfitication system," *Hymenoptera: evolution, biodiversity and biological-control*, pp. 303–311, 2000.

[22] N. MacLeod, M. Benfield, and P. Culverhouse, "Time to automate identification," *Nature*, vol. 467, no. 7312, pp. 154–55, 2010. [Online]. Available: http://www.nature.com/nature/journal/v467/n7312/full/467154a.html?type=access_denied

[23] C. Darwin, *On the Origin of Species.* England: Dover Publications, 1859.

[24] B. Li and L. Tusheng, "Comments on coevolution in genetic algorithms (in chinsese)," *Computer Science*, vol. 36, no. 4, pp. 34–63, 2009.

[25] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Boston, Massachusetts: MIT Press, 1992.

[26] M. J. W. Lawrence J. Fogel, Alvin J. Owens, *Artificial Intelligence through Simulated Evolution.* Chichester, UK: Wiley, 1966.

[27] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Stuttgart: Fromman-Hozlboog Verlag, 1994.

[28] J. Koza, *Genetic Programming.* Cambridge MA: MIT Press, 1992.

[29] C. Rosin and R. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, no. 10, pp. 1–29, 1997.

[30] R. Dawkins and J. R. Krebs, "Arms races between and within species," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 205, no. 1161, pp. 489–511, 1979. [Online]. Available: http://rspb.royalsocietypublishing.org/content/205/1161/489.abstract

[31] J. Cartlidge and S. Bullock, "Combating coevolutionary disengagement by reducing parasite virulence," *Evolutionary Computation*, vol. 12, no. 2, pp. 193–222, 2004.

[32] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," *Bibliometrics*, vol. 155, no. 18, pp. 1–5, 1993.

[33] L. Panait and S. Luke, "A comparative study of two competitive fitness functions," in *Proceedings of the Genetic and Evolutionary Computation Conference.* Boston, Massachusetts: MIT Press, 2002, pp. 567–573.

[34] T. Tan and J. Teo, "Competitive coevolution with k-random opponents for pareto multiobjective optimization," in *Natural Computation, Third International Conference on*, 2007, pp. 63 – 67. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.38.3029

[35] D. B. Fogel, "The advantages of evolutionary computation," in *Biocomputing and Emergent Computation: Proceedings of BCEC97*. World Scientific Press, 1997, pp. 1–11.

[36] H. GS, L. JD, and L. DS, "Computer-automated evolution of an x-band antenna for nasa's space technology 5 mission," *Evolutionary Computation*, vol. 19, no. 1, pp. 1–23, 2011.

[37] O. E. David, H. J. van den Herik, M. Koppel, and N. S. Netanyahu, "Genetic algorithms for evolving computer chess programs," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 5, pp. 779–789, 2014.

[38] S. A. Kauffman, "Escaping the red queen effect," *The McKinsey Quarterly*, no. 1, p. 118, 1995.

[39] G. Gutin, A. Yeo, and A. Zverovich, "Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the {TSP}," *Discrete Applied Mathematics*, vol. 117, no. 13, pp. 81 – 86, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X01001950

[40] C. Wang, S. Yu, W. Chen, and C. Sun, "Highly efficient light-trapping structure design inspired by natural evolution," *Sci. Rep.*, vol. 3, no. 1, pp. 1–7, 2013.

[41] R. Bellman, *Dynamic Progamming and Lagrange Multipliers*. Princeton, NJ, USA: Princeton University Press, 1957.

[42] J. J. E. Dennis and J. J. Mor, "Quasi-newton methods, motivation and theory," *SIAM Review*, vol. 19, no. 1, pp. 46–89, 1977.

[43] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Pittsburgh, PA, USA, Tech. Rep., 1994.

[44] N. Hansen, S. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, March 2003.

[45] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[46] K. J. R. and J. P. Rice, "Automatic programming of robots using genetic programming," in *AAAI*. MIT Press, 1992, pp. 1–6.

[47] R. A. Brooks, "Artificial life and real robots," in *Proceedings of the First European Conference on Artificial Life*. MIT Press, 1992, pp. 3–10.

[48] D. Floreano and S. Nolfi, "Adaptive behavior in competing co-evolving species," in *The 4th European Conference on Artificial Life*. MIT Press, 1997, pp. 378–387.

[49] D. Floreano, P. Drr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008. [Online]. Available: http://dx.doi.org/10.1007/s12065-007-0002-4

[50] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ec02

[51] B. D.M. and O. C., "Understanding evolutionary potential in virtual cpu instruction set architectures," *PLoS ONE*, vol. 8, no. 12, p. e83242, 2013. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ec02

[52] B. Batut, D. P. Parsons, S. Fischer, G. Beslon, and C. Knibbe, "In silico experimental evolution: a tool to test evolutionary scenarios," in *Proceedings of the Eleventh Annual Research in Computational Molecular Biology (RECOMB) Satellite Workshop on Comparative Genomics*. BioMed Central Ltd, 2013, pp. 1–6.

[53] J.-M. Montanier and N. Bredeche, "Surviving the Tragedy of Commons: Emergence of Altruism in a Population of Evolving Autonomous Agents," in *European Conference on Artificial Life*, Paris, France, Aug. 2011. [Online]. Available: https://hal.inria.fr/inria-00601776

[54] W. M, F. D, and K. L, "A quantitative test of hamilton's rule for the evolution of altruism," *PLoS Biology*, vol. 9, no. 5, p. e1000615, 2011.

[55] D. Floreano, S. Mitri, S. Magnenat, and L. Keller, "Evolutionary conditions for the emergence of communication in robots," *Current Biology*, vol. 17, no. 6,

pp. 514 – 519, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960982207009281

[56] A. JE and B. JC, "Environmental influence on the evolution of morphological complexity in machines," *PLoS Computational Biology*, vol. 10, no. 1, p. e1003399, 2014.

[57] D. Cliff and G. F. Miller, "Co-evolution of pursuit and evasion ii: Simulation methods and results," *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, vol. 92, no. 2, pp. 101–106, 1995.

[58] D. Floreano, "Evolutionary robotics in behavior engineering and artificial life," in *Evolutionary Robotics: From Intelligent Robots to Artificial Life. Applied AI Systems, 1998. Evolutionary Robotics Symposium.* AAI Books, 1998.

[59] S. Koos, J.-B. Mouret, and S. Doncieux, "The transferability approach: Crossing the reality gap in evolutionary robotics," *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 1, pp. 122–145, Feb 2013.

[60] S. Koos, A. Cully, and J. Mouret, "Fast damage recovery in robotics with the t-resilience algorithm," *CoRR*, vol. abs/1302.0386, 2013. [Online]. Available: http://arxiv.org/abs/1302.0386

[61] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Trans. Syst., Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 3, pp. 396–407, 1996.

[62] R. Watson, S. Ficiei, and J. Pollack, "Embodied evolution: embodying an evolutionary algorithm in a population of robots," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 1, 1999, pp. –342 Vol. 1.

[63] A. Eiben, E. Haasdijk, and N. Bredeche, "Embodied, On-line, On-board Evolution for Autonomous Robotics," in *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution.*, ser. Series: Cognitive Systems Monographs, S. K. E. P. Levi, Ed. Springer, 2010, vol. 7, pp. 361–382. [Online]. Available: https://hal.inria.fr/inria-00531455

[64] A. Eiben, S. Kernbach, and E. Haasdijk, "Embodied artificial evolution," *Evolutionary Intelligence*, vol. 5, no. 4, pp. 261–272, 2012. [Online]. Available: http://dx.doi.org/10.1007/s12065-012-0071-x

[65] A. E. Eiben and J. Smith, “From evolutionary computation to the evolution of things,” *Nature*, vol. 521, no. 7553, pp. 467–482, 2015.

[66] J. R. Tumbleston, D. Shirvanyants, N. Ermoshkin, R. Janusziewicz, A. R. Johnson, D. Kelly, K. Chen, R. Pinschmidt, J. P. Rolland, A. Ermoshkin, E. T. Samulski, and J. M. DeSimone, “Continuous liquid interface production of 3d objects,” *Science*, vol. 347, no. 6228, pp. 1349–1352, 2015. [Online]. Available: http://www.sciencemag.org/content/347/6228/1349.abstract

[67] L. Ljung, “System identification: Theory for the user,” *Englewood Cliffs, NJ: Prentice-Hall*, 1999.

[68] D. B. Fogel, *System identification through simulated evolution: a machine learning approach to modeling.* Needham, MA, USA: Ginn Press, 1991.

[69] S. A. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains.* Hoboken, NJ, USA: Wiley-Blackwell, 2013.

[70] E. J. Vladislavleva, G. F. Smits, and D. Den Hertog, “Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming,” *Trans. Evol. Comp*, vol. 13, no. 2, pp. 333–349, Apr. 2009. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2008.926486

[71] J. Bongard and H. Lipson, “Nonlinear system identification using coevolution of models and tests,” *IEEE Trans. Evol. Comput.*, vol. 9, no. 4, pp. 361–384, 2005.

[72] ——, “Automated damage diagnosis and recovery for remote robotics,” in *Proc. 2004 IEEE Int. Conf. Robot. and Autom.* IEEE Computer Society Press, New Orleans, LA, 2004, pp. 3545–3550.

[73] ——, “Automated robot function recovery after unanticipated failure or environmental change using a minimum of hardware trials,” in *Proc. 2004 NASA/DoD Conf. Evolvable Hardware.* IEEE Computer Society Press, Los Alamitos, CA, 2004, pp. 169–176.

[74] S. Koos, J. Mouret, and S. Doncieux, “Automatic system identification based on coevolution of models and tests,” in *Proc. 2009 IEEE Congr. Evol. Computation.* IEEE Press, Trondheim, Norway, 2009, pp. 560–567.

[75] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *PNAS*, vol. 104, no. 24, pp. 9943–9948, 2007.

[76] M. Mirmomeni and W. Punch, "Co-evolving data driven models and test data sets with the application to forecast chaotic time series," in *Proc. 2011 IEEE Congr. Evol. Comput.* IEEE Press, New Orleans, LA, USA, 2011, pp. 14–20.

[77] D. Le Ly and H. Lipson, "Optimal experiment design for coevolutionary active learning," *IEEE Trans. Evol. Computation*, vol. 18, no. 3, pp. 394–404, 2014.

[78] B. Kouchmeshky, W. Aquino, J. C. Bongard, and H. Lipson, "Co-evolutionary algorithm for structural damage identification using minimal physical testing," *International Journal for Numerical Methods in Engineering*, vol. 69, no. 5, pp. 1085–1107, 2007. [Online]. Available: http://dx.doi.org/10.1002/nme.1803

[79] M. Mirmomeni and W. Punch, "Co-evolving data driven models and test data sets with the application to forecast chaotic time series," in *2011 IEEE Congress on Evolutionary Computation.* Auburn University, New Orleans, LA, 2011, pp. 14 –20.

[80] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Sci.*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[81] S. Koos, J. B. Mouret, and S. Doncieux, "The transferability approach: Crossing the reality gap in evolutionary robotics," *IEEE Trans. Evol. Computation*, vol. 17, no. 1, pp. 122–145, Feb 2013.

[82] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[83] P. J. O'Dowd, M. Studley, and A. F. T. Winfield, "The distributed co-evolution of an on-board simulator and controller for swarm robot behaviours," *Evol. Intell.*, vol. 7, no. 2, pp. 95–106, 2014.

[84] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: the use of simulation in evolutionary robotics," in *Advances in Artificial Life: Proc. 3rd European Conf. Artificial Life.* Springer-Verlag, 1995, pp. 704–720.

[85] B. Hedwig and J. F. A. Poulet, "Complex auditory behaviour emerges from simple reactive steering," *Nature*, vol. 430, no. 7001, pp. 781–785, 2004.

[86] E. Baird, M. J. Byrne, J. Smolka, E. J. Warrant, and M. Dacke, "The dung beetle dance: An orientation behaviour?" *PLoS ONE*, vol. 7, no. 1, p. e30211, 01 2012. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0030211

[87] M. D. M. Byrne, "Visual cues used by ball-rolling dung beetles for orientation," *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, vol. 189, no. 6, pp. 411–418, 2003.

[88] E. G. Matthews, "Observations on the ball-rolling behavior of canthon pilularius," *Psyche*, pp. 75–93, 1963.

[89] S. Camazine, J.-L. Deneubourg, N. R. Franks, *et al.*, *Self-Organization in Biological Systems.* Princeton, NJ: Princeton University Press, 2001.

[90] S. Garnier, J. Gautrais, and G. Theraulaz, "The biological principles of swarm intelligence," *Swarm Intelligence*, vol. 1, no. 1, pp. 3–31, 2007. [Online]. Available: http://dx.doi.org/10.1007/s11721-007-0004-y

[91] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[92] R. Jeanson, C. Rivault, J.-L. Deneubourg, S. Blanco, R. Fournier, C. Jost, and G. Theraulaz, "Self-organized aggregation in cockroaches," *Animal Behaviour*, vol. 69, no. 1, pp. 169 – 180, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003347204002428

[93] C. R. Carroll and D. H. Janzen, "Ecology of foraging by ants," *Annu. Review of Ecology and Systematics*, vol. 4, pp. 231–257, 1973.

[94] J. E. Lloyd, "Bioluminescent communication in insects," *Annual Review of Entomology*, vol. 16, pp. 97–122, 1971.

[95] O. H. Bruinsma, "An analysis of building behaviour of the termite macrotermes subhyalinus (rambur)," Ph.D. dissertation, Wageningen University, Wageningen, The Netherlands, 1979.

[96] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.

[97] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.

[98] O. Holland and C. Melhuish, “Stigmergy, self-organization, and sorting in collective robotics,” *Artificial Life*, vol. 5, no. 2, pp. 173–202, 1999.

[99] G. Di Caro and M. Dorigo, “Antnet: Distributed stigmergetic control for communications networks,” *J. Artif. Int. Res.*, vol. 9, no. 1, pp. 317–365, Dec. 1998. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622797.1622806

[100] K. Socha, “Aco for continuous and mixed-variable optimization,” in *Ant Colony Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. Sttzle, Eds. Springer Berlin Heidelberg, 2004, vol. 3172, pp. 25–36. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-28646-2_3

[101] J. Bjerknes and A. T. Winfield, “On fault tolerance and scalability of swarm robotic systems,” in *Distributed Autonomous Robotic Systems*, ser. Springer Tracts in Advanced Robotics. Springer, Berlin, Heidelberg, 2013, vol. 83, pp. 431–444.

[102] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Gros, “Occlusion-based cooperative transport with a swarm of miniature mobile robots,” *Robotics, IEEE Transactions on*, vol. 31, no. 2, pp. 307–321, April 2015.

[103] M. Gauci, J. Chen, T. Dodd, and R. Groß, “Evolving aggregation behaviors in multi-robot systems with binary sensors,” in *Distributed Autonomous Robotic Systems*, ser. Springer Tracts in Advanced Robotics. Springer, Berlin, Heidelberg, 2014, vol. 104, pp. 355–367.

[104] E. ahin, “Swarm robotics: From sources of inspiration to domains of application,” in *Swarm Robotics*, ser. Lecture Notes in Computer Science, E. ahin and W. Spears, Eds. Springer Berlin Heidelberg, 2005, vol. 3342, pp. 10–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30552-1_2

[105] B. Gerkey and M. Mataric, “Sold!: auction methods for multirobot coordination,” *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 758–768, Oct 2002.

[106] A. F. T. Winfield, "Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots," in *Distributed Autonomous Robotic Systems 4*, L. Parker, G. Bekey, and J. Barhen, Eds. Springer Japan, 2000, pp. 273–282. [Online]. Available: http://dx.doi.org/10.1007/978-4-431-67919-6_26

[107] V. Trianni, R. Gro, T. Labella, E. ahin, and M. Dorigo, "Evolving aggregation behaviors in a swarm of robots," in *Advances in Artificial Life*, ser. Lecture Notes in Computer Science, W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, and J. Kim, Eds. Springer Berlin Heidelberg, 2003, vol. 2801, pp. 865–874. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-39432-7_93

[108] M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß, "Self-organized aggregation without computation," *The Int. J. of Robot. Research*, vol. 33, no. 8, pp. 1145–1161, 2014.

[109] S. Garnier, C. Jost, J. Gautrais, M. Asadpour, G. Caprari, R. Jeanson, A. Grimal, and G. Theraulaz, "The embodiment of cockroach aggregation behavior in a group of micro-robots," *Artificial Life*, vol. 14, no. 4, pp. 387–408, Oct. 2008. [Online]. Available: http://dx.doi.org/10.1162/artl.2008.14.4.14400

[110] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*. Springer, 2002, pp. 299–308.

[111] J. McLurkin and J. Smith, "Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots," in *in 7th International Symposium on Distributed Autonomous Robotic Systems (DARS*. Citeseer, 2004.

[112] K. Fujibayashi, S. Murata, K. Sugawara, and M. Yamamura, "Self-organizing formation algorithm for active elements," in *Reliable Distributed Systems, 2002. Proceedings. 21st IEEE Symposium on*, 2002, pp. 416–421.

[113] J. Chen, M. Gauci, M. J. Price, and R. Groß, "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 - Volume 1*, ser. AAMAS '12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 163–170. [Online]. Available: http://dl.acm.org/citation.cfm?id=2343576.2343599

[114] A. Turgut, H. elikkanat, F. Gke, and E. ahin, "Self-organized flocking in mobile robot swarms," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 97–120, 2008. [Online]. Available: http://dx.doi.org/10.1007/s11721-008-0016-2

[115] E. B. C.R. Kube, "Collective robotics: from social insects to robots," *Adaptive Behavior*, vol. 2, no. 2, pp. 189–218, 1993.

[116] C. Kube and E. Bonabeau, "Cooperative transport by ants and robots," *Robotics and Autonomous Systems*, vol. 30, no. 12, pp. 85 – 101, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889099000664

[117] R. Gross and M. Dorigo, "Towards group transport by swarms of robots," *Int. J. Bio-Inspired Comput.*, vol. 1, no. 1/2, pp. 1–13, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1504/IJBIC.2009.022770

[118] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[119] G. S. Fraenkel and D. L. Gunn, *The Orientation of Animals: Kineses, Taxes, and Compass Reactions.* New York: Dover Publications, 1961.

[120] S. D. Sulkin, "Larval orientation mechanisms: The power of controlled experiments," *Ophelia*, vol. 32, no. 1-2, pp. 49–62, 1990.

[121] I. Rano, "A steering taxis model and the qualitative analysis of its trajectories," *Adaptive Behaviour*, vol. 17, no. 3, pp. 197–211, 2009.

[122] S. Camazine, *Self-organization in biological systems.* Princeton University Press, 2003.

[123] B. Webb, "What does robotics offer animal behaviour?" *Animal Behaviour*, vol. 60, no. 5, pp. 545 – 558, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003347200915148

[124] J.-A. Meyer and A. Guillot, "Biologically inspired robots," in *Springer Handbook of Robot.*, ser. Springer Handbooks, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg, Germany: Springer, 2008, pp. 1395–1422.

[125] B. Webb, "Using robots to model animals: a cricket test," *Robotics and Autonomous Systems*, vol. 16, no. 2134, pp. 117 – 134, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0921889095000445

[126] A. Popov and V. Shuvalov, “Phonotactic behavior of crickets,” *J. of Comparative Physiology*, vol. 119, no. 1, pp. 111–126, 1977.

[127] A. M. Farah and T. Duckett, “Reactive localisation of an odour source by a learning mobile robot,” in *In Proceedings of the Second Swedish Workshop on Autonomous Robotics*. SWAR Stockholm, Sweden, 2002, pp. 29–38.

[128] A. Lilienthal and T. Duckett, “Experimental analysis of smelling braitenberg vehicles,” in *In Proceedings of the ieee international conference on advanced robotics*. Coimbra, Portugal, 2003, pp. 58–63.

[129] T. Balch, F. Dellaert, A. Feldman, A. Guillory, C. Isbell, Z. Khan, S. Pratt, A. Stein, and H. Wilde, “How multirobot systems research will accelerate our understanding of social animal behavior,” *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1445 –1463, 2006.

[130] J. Chappell and S. Thorpe, “Ai-inspired biology: Does ai have something to contribute to biology?” *Proceedings of the International Symposium on AI Inspired Biology: A Symposium at the AISB 2010 Convention, Leicester, UK*, 2010.

[131] J. Faria, J. Dyer, R. Clément, *et al.*, “A novel method for investigating the collective behaviour of fish: Introducing ‘robofish’,” *Behavioral Ecology and Sociobiology*, vol. 64, no. 8, pp. 1211–1218, 2010.

[132] J. Halloy, F. Mondada, S. Kernbach, *et al.*, “Towards bio-hybrid systems made of social animals and robots,” in *Biomimetic and Biohybrid Systems*, ser. Lecture Notes in Comput. Sci. Springer, Berlin, Heidelberg, Germany, 2013, vol. 8064, pp. 384–386.

[133] J. Halloy, G. Sempo1, G. Caprari, *et al.*, “Social integration of robots into groups of cockroaches to control self-organized choices,” *Sci.*, vol. 318, no. 5853, pp. 1155–1158, 2007.

[134] T. Schmickl, S. Bogdan, L. Correia, *et al.*, “Assisi: Mixing animals with robots in a hybrid society,” in *Biomimetic and Biohybrid Systems*, ser. Lecture Notes in Comput. Sci. Springer, Berlin, Heidelberg, Germany, 2013, vol. 8064, pp. 441–443.

[135] R. Vaughan, N. Sumpter, J. Henderson, *et al.*, “Experiments in automatic flock control,” *Robot. and Autonomous Syst.*, vol. 31, no. 1, pp. 109–117, 2000.

[136] J. Krause, A. F. Winfield, and J.-L. Deneubourg, "Interactive robots in experimental biology," *Trends in Ecology and Evolution*, vol. 26, no. 7, pp. 369 –375, 2011.

[137] S. G. Halloy J., "Social integration of robots into groups of cockroaches to control self-organized choices," *Science*, vol. 318, no. 5853, pp. 1155–1158, 2007. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/sci;318/5853/1155

[138] J. Krause, A. F. Winfield, and J.-L. Deneubourg, "Interactive robots in experimental biology," *Trends in Ecology and Evolution*, vol. 26, no. 7, pp. 369 – 375, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169534711000851

[139] R. Vaughan, N. Sumpter, A. Frost, and S. Cameron, "Robot sheepdog project achieves automatic flock control," *The fourth international conference on Autonomous agents*, pp. 489–493, 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.38.3029

[140] A. Gribovskiy, J. Halloy, J.-L. Deneubourg, H. Bleuler, and F. Mondada, "Towards mixed societies of chickens and robots," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on.* Boston, Massachusetts: MIT press, 2010, pp. 4722 –4728.

[141] V. Kopman, J. Laut, G. Polverino, *et al.*, "Closed-loop control of zebrafish response using a bioinspired robotic-fish in a preference test," *J. of The Roy. Soc. Interface*, vol. 10, no. 78, pp. 1–8, 2013.