The University Of Sheffield.

Automated Reverse Engineering of Agent Behaviors

Wei Li

A thesis submitted for the degree of
Doctor of Philosophy

Department of Automatic Control and Systems Engineering
The University of Sheffield

30th September 2015

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 through observation. A replica, which resembles the agents under investigation in terms of behavioral capabilities, is mixed into the group. The models are executed on the replica. This case study is conducted with swarms of both simulated and physical robots. 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 that the agent responds to. In the case study of inferring deterministic behavior, the machine can construct static patterns of stimuli that facilitate the learning process. In the case study of inferring stochastic behavior, the machine needs to interact with the agent on the fly through dynamically changing patterns of stimuli. This allows the machine to explore the agent's hidden information and thus reveal its entire behavioral repertoire. This interactive approach proves superior to learning only through observation.

Acknowledgments

I consider the process of pursuing my PhD as a journey to learn not only about science, but also about life. Firstly, I would like to thank my supervisors, Dr. Roderich Groß and Prof. Stephen A. Billings for their support in this journey. The special thanks would be given to Dr. Roderich Groß. He helped me from toddling to walking on the pathway in this research field by motivating, teaching and passing his professional experience to me without any reservation. He is always very patient to discuss with me and give me professional suggestions. His passion for pursuing science and rigorous attitude in research have deeply influenced me.

Second, many people have contributed to this journey, especially people in the Natural Robotics Lab: Melvin Gauci, Jianing Chen, Yuri K. Lopes, Christopher Parrott, Fernando Perez Diaz, Stefan Trenkwalder, Gabriel Kapellmann Zafra, Matthew Doyle, and Shen-Chiang Chen, who make the lab a warm, supportive and fun environment. Special thanks to Melvin Gauci and Jianing Chen. To Melvin, I considered myself very lucky that I met him from the start of this unforgettable journey. I have learned much from him on doing research and life. We not only had many collaborations on research, but we also shared the experience outside science. To Jianing, I appreciate his help for sharing his knowledge with me and providing valuable feedback when I did my experiments.

Outside the research group, there are also a number of people who helped me during my stay in Sheffield, especially Xiao Chen and Yuzhu Guo. This journey would not be completed without their encouragement.

Finally, I would like to thank my family. To my parents, thank you for giving me a free environment to grow up and always supporting me without any hesitation. To my wife, Yifei, thank you for always accompanying and encouraging me, which makes me maintain a positive attitude for life.

Contents

1 Reverse Engineering Swarm Behaviors Through Turing Learning

As mentioned in Chapter **??**, swarm behaviors are emergent behaviors that arise from the interactions of a number of simple individuals in a group [17]. Researchers in artificial intelligence and swarm robotics use simulated agents or physical robots to mimic and understand the swarm behaviors observed in nature, such as foraging, aggregation, and flocking. This approach is what we refer to as understanding from synthesis. It has been used to validate models of swarm behaviors, provide insights, or even generate new hypotheses [165]. The design of controllers for the simulated agents or physical robots usually adheres to similar conditions to those found in biological swarm systems. For example, the controllers are distributed, that is, there is no central control for the swarm; the structure of the controllers is relatively simple.

In this chapter, we demonstrate how the proposed system identification (coevolutionary) method allows a machine to infer the behavioral rules of a group of homogeneous agents in an autonomous manner. A replica, which resembles the agents under investigation in terms of behavioral capabilities, is mixed into the group. The coevolutionary algorithm consists of two competitive populations: one of *models*, to be executed on the replica, and the other of *classifiers*. The classifiers observe the motion of an individual[1] in the swarm for a fixed time interval. They are not, however, provided with the individual's sensory information. Based on the individual's motion data, a classifier outputs a Boolean value indicating whether the individual is believed to be an agent or replica. The classifier gets a reward if and only if it makes the correct judgment. The fitness of the classifiers thus depends solely on their ability to discriminate between the agents

[1]Note that 'individual/robot' in the context of swarm behaviors refers to a single unit. It could refer to an agent under investigation or a replica executing a model. However, in the context of evolutionary algorithms (see Section 1.1.1.3), 'individual' refers to a chromosome.

and the replica. Conversely, the fitness of the models depends solely on their ability to 'trick' the classifiers into categorizing them as an agent. Consequently, our method does not rely on predefined metrics for measuring the similarity of behavior between models and agents; rather, the metrics (classifiers) are produced automatically in the learning process. Our method is inspired by the Turing test [174, 175], which machines can pass if behaving indistinguishably from humans. Similarly, the models, which evolve, can pass the tests by the coevolving classifiers if behaving indistinguishably from the agents. We hence call our method *Turing Learning*.

To validate the *Turing Learning* method, we present two case studies on canonical problems in swarm robotics: self-organized aggregation [21] and object clustering [22]. We show that observing individual motion is sufficient to guide the learning of these collective behaviors. In this chapter, we only present the simulation results. A real-world validation using physical robots based on one of the case studies will be presented in Chapter **??**.

This chapter is organized as follows. Section 1.1 describes the implementation of *Turing Learning* and the two swarm behaviors under investigation. Section 1.2 introduces the simulation platform and setups for performing coevolution runs. Section 1.3 presents the results obtained from the two case studies. In particular, Section 1.3.1 analyzes the evolution of models, objectively measuring their quality in terms of local and global behaviors. Section 1.3.2 investigates the coevolutionary dynamics. Section 1.3.3 investigates the evolution of classifiers, showing how to construct a robust classifier system to potentially detect abnormal behaviors in the swarm. Section 1.3.4 studies the effect of observing only a subset of agents in the swarm. Section 1.3.5 presents a study where an aspect of the agents' morphology (their field of view) and brain (controller) are inferred simultaneously. Section 1.3.6 shows the results of using *Turing Learning* to learn other swarm behaviors. Section 1.3.7 presents a study showing the method's sensitivity to noise. Section 1.4 summaries the findings in this chapter.

1.1 Methodology

In this section, we present the *Turing Learning* method and two case studies: self-organized aggregation [21] and object clustering [22]. In both case studies, individuals

execute simple behavioral rules that lead to meaningful emergent behaviors on a global level.

1.1.1 Turing Learning

Turing Learning uses a coevolutionary algorithm that comprises two populations: one of models, and one of classifiers. These populations coevolve competitively. In the following, we describe the models, classifiers, optimization algorithm, fitness calculation method and termination criterion.

1.1.1.1 Models

We assume to have a replica that has actuation and sensing abilities that are equivalent to those of the agents under investigation. In other words, the replica must be able to produce behavior that—to an external observer (classifier)—is indistinguishable to that of the agent. For example, if the observer tracks the position of a bird in three dimensions (3D), the replica (which could be artificial or real) must be able to produce 3D trajectory data. There can be one or more replicas. In Chapters 1 and **??**, the replica(s) will be mixed into a group of homogeneous agents. They should therefore be perceived by the agents as con-specifics [165]. In Chapter 2, a single replica is studied in isolation.

Models are executed on replicas. The models can be represented explicitly (e.g., parameters) or implicitly (e.g., artificial neural networks).

1.1.1.2 Classifiers

The classifier can be any algorithm that takes a sequence of data about the observed individual as input, and outputs a decision (i.e., whether the observed individual is believed to be an agent or replica).

We represent the classifier as a recurrent Elman neural network [176], which is shown in Figure 1.1. The network has i inputs, h hidden neurons and one output neuron.

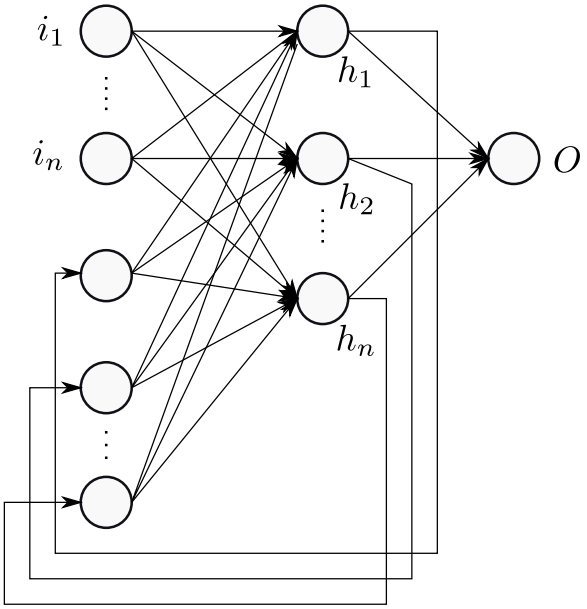


Figure 1.1: The plot shows the structure of a classifier, which is a recurrent Elman neural network, with i inputs, h hidden neurons, and one output neuron (O). O is used for making a judgment. Two bias neurons (which are not shown) with a constant input of 1.0 are connected to each neuron of the hidden and output layers. See text for details.

Each neuron of the hidden and output layers has a bias. The network has a total of $(i+1)h+h^2+(h+1)$ parameters, which all assume values in $\mathbb{R}$. The activation function used in the hidden and the output neurons is the logistic sigmoid function, which has the range $(0, 1)$ and is defined as:

$$\operatorname{sig} x = \frac{1}{1+e^{-x}} \quad \forall x \in \mathbb{R}. \tag{1.1}$$

The classifier makes judgments about all individuals—both agents and replica. It observes each individual over a set period of time (i.e., a trial). The final value of its output neuron, O, is used to make the judgment: the classifier judges the individual to be a replica if $O < 0.5$, and an agent otherwise. The classifier's memory (value of hidden neurons) is reset after each judgment.

1.1.1.3 Optimization Algorithm

The optimization of models and classifiers is realized using evolutionary algorithms. In this thesis, we use a $(\mu+\lambda)$ evolution strategy with self-adaptive mutation strengths [177, 109] to optimize either population. The optimization algorithm can be thought of consisting of two sub-algorithms: one for the model population, and another for the classifier population. The sub-algorithms do not interact with each other except for the fitness calculation step (described in Section 1.1.1.4). In each sub-algorithm, the value of μ and λ is equal, which is half of the population size. The implementation of the evolutionary algorithm is detailed as follows.

In this algorithm, an individual is a 2-tuple, $\mathbf{a} = (\mathbf{x}, \boldsymbol{\sigma})$, where $\mathbf{x} \in \mathbb{R}^n$ represents objective parameters, and $\boldsymbol{\sigma} \in (0, \infty)^n$ represents mutation strengths. The i-th mutation strength in $\boldsymbol{\sigma}$ corresponds to the i-th element in $\mathbf{x}$.

Each generation g comprises a population of μ individuals:

$$\mathcal{P}^{(g)} = \left\{ \mathbf{a}_1^{(g)}, \mathbf{a}_2^{(g)}, \ldots, \mathbf{a}_\mu^{(g)} \right\}.$$

In the population of the first generation, $\mathcal{P}^{(0)}$, all the objective parameters are initialized to 0.0 and all the mutation strengths are initialized to 1.0. Thereafter, in every generation g, the μ parent individuals are first used to create λ offspring individuals by recombination. For the generation of each recombined individual $\mathbf{a}_k'^{(g)}$, $k \in \{1, 2, \ldots, \lambda\}$, two individuals are chosen randomly, with replacement, from the parent population: $\mathbf{a}_\chi^{(g)}$ and $\mathbf{a}_\psi^{(g)}$, where $\chi, \psi \in \{1, 2, \ldots, \mu\}$. The intermediate population, $\mathcal{P}'^{(g)}$, is produced using discrete and intermediate recombination, which generates the objective parameters and the mutation strengths of the recombined individual, respectively:

$$x_{k,i}'^{(g)} = x_{\chi,i}^{(g)} \quad \text{OR} \quad x_{\psi,i}^{(g)}, \tag{1.2}$$

$$\sigma_{k,i}'^{(g)} = \left(\sigma_{\chi,i}^{(g)} + \sigma_{\psi,i}^{(g)} \right) / 2, \tag{1.3}$$

where $i \in \{1, 2, \ldots, n\}$ is indexing the elements within the vectors and, in Equation 1.2, the selection is performed randomly and with equal probability.

Each of the λ recombined individuals is then mutated in order to obtain the final offspring

population, $\mathcal{P}''^{(g)}$. This is done according to:

$$\sigma''^{(g)}_{k,i} = \sigma'^{(g)}_{k,i} \exp\left(\tau' \mathcal{N}_k\left(0,1\right) + \tau \mathcal{N}_{k,i}\left(0,1\right)\right), \tag{1.4}$$

$$x''^{(g)}_{k,i} = x'^{(g)}_{k,i} + \sigma''^{(g)}_{k,i} \mathcal{N}_{k,i}\left(0,1\right), \tag{1.5}$$

for all $\{k,i\}$, where $k \in \{1,2,\ldots,\lambda\}$ is indexing the individuals within the population and $i \in \{1,2,\ldots,n\}$ is indexing the elements within the vectors. Equation 1.4 generates the perturbed mutation strength from the original one according to a log-normal distribution. Equation 1.5 mutates the objective parameter according to a normal distribution having the perturbed mutation strength as its deviation. In Equation 1.4, $\mathcal{N}_k\left(0,1\right)$ and $\mathcal{N}_{k,i}\left(0,1\right)$ are both random numbers generated from a standard normal distribution; however, the former is generated once for each individual (i.e. for each value of k), while the latter is generated separately for each element within each individual (i.e. for each combination of k and i). The parameters τ' and τ determine the learning rates of the mutation strengths, and are set as $\tau' = 1/2\sqrt{2n}$, $\tau = 1/2\sqrt{2\sqrt{n}}$ (similar to [178]), where n corresponds to the population size.

Once the offspring population has been generated, the μ individuals with the highest fitness from the combined population, $\mathcal{P}^{(g)} \cup \mathcal{P}''^{(g)}$ (which contains $\mu+\lambda$ individuals), are selected as the parents to form the population of the next generation, $\mathcal{P}^{(g+1)}$. Individuals with an equal fitness have an equal chance of being selected.

1.1.1.4 Fitness Calculation

Let the population sizes for the models and classifiers in the coevolution be M and C, respectively. Let the number of replicas and agents in a trial be n_r and n_a, respectively. n_t trials are conducted for a model in each generation; throughout this thesis, we assume $n_t = 1$.

In our thesis, whenever multiple replicas are used, each of them executes a different model. The fitness of each model in a trial is determined by each of the C classifiers in the competing population. For every classifier that wrongly judges the model as an agent, the model's fitness increases by 1. After all evaluations, the model's fitness takes a value in $\{0,1,2,\ldots,C\}$. This value is then normalized to $[0,1]$.

The fitness of each classifier in a trial is determined by its judgments for the n_r replicas (each executing a different model) and n_a agents. For each correct judgment of the replica, the classifier's fitness increases by $\frac{1}{2n_r}$; for each correct judgment of the agent, the classifier's fitness increases by $\frac{1}{2n_a}$. Therefore, the fitness of each classifier in a trial is in range $[0, 1]$. $\left\lceil \frac{M}{n_r} \right\rceil$ trials[2] are conducted in each generation, and the fitness value of each classifier is then normalized to $[0, 1]$.

1.1.1.5 Termination Criterion

The coevolutionary algorithm stops after running for a fixed number of generations.

1.1.2 Case Studies

1.1.2.1 Problem Formulation

The agents used in the case studies of Chapters 1 and **??** are embodied and move in a two-dimensional, continuous space. The agents' embodiment is based on the e-puck [179], which is a miniature, differential-wheeled robot. Figure 1.2 shows an e-puck robot used in the physical experiments in Chapter **??**.

Each agent is equipped with a line-of-sight sensor that it can use to detect the item (e.g., the background, other agents or objects [21], [22]) in front of it.

The swarm behaviors investigated in this thesis use a reactive control architecture, as found in many biological systems[3]. The motion of each agent solely depends on the state of its line-of-sight sensor (I). Each possible sensor state, $I \in \{0, 1, \cdots, n-1\}$, is mapped onto a pair of predefined velocities for the left and right wheels, $(v_{\ell I}, v_{rI})$. $v_{\ell I}, v_{rI} \in [-1, 1]$ represent the normalized left and right wheel velocities, respectively, where 1 (-1) corresponds to the wheel rotating forwards (backwards) with maximum velocity. Given n sensor states, any reactive behavior can thus be represented using

[2] We suggest choosing n_r to be a factor of M.

[3] For example, researchers have found that the complex auditory orientation behavior of female crickets is derived from simple reactive motor responses to specific sound pulses [120].

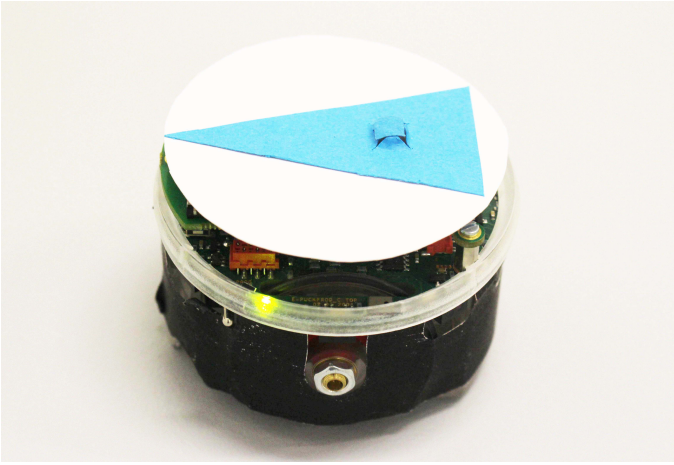


Figure 1.2: An e-puck robot fitted with a black 'skirt' and a top marker for motion tracking.

$2n$ system parameters. In the remainder of this thesis, we describe the corresponding controllers by writing the $2n$ parameters as a tuple in the following order:

$$\mathbf{p} = (v_{\ell 0}, v_{r0}, v_{\ell 1}, v_{r1}, \cdots, v_{\ell(n-1)}, v_{r(n-1)}). \tag{1.6}$$

We assume that the replica has the same differential drive and line-of-sight sensor[4] as the agents. The system identification task is thus to infer the control parameters in Equation (1.6). This explicit representation makes it possible for us to objectively measure the quality of the obtained models in the post-evaluation analysis (as discussed in the results section). To make the evolution more challenging, the search space for the algorithm in simulation is unbounded. That is, the model parameters are unconstrained, and the replica can move with arbitrary speed.

The classifier does not have any prior knowledge about the individual under investigation. It is fed with the sequence of motion data of the individual. It has two input neurons ($i = 2$), five hidden neurons ($h = 5$) and one output neuron. The input neurons represent the linear speed (v) and angular speed (ω) of the individual. They are obtained by tracking the positions and orientations of individuals. In simulation, the tracking is

[4]In Section 1.3.5, we show that this assumption can be relaxed by also evolving some aspect of the agent's morphology.

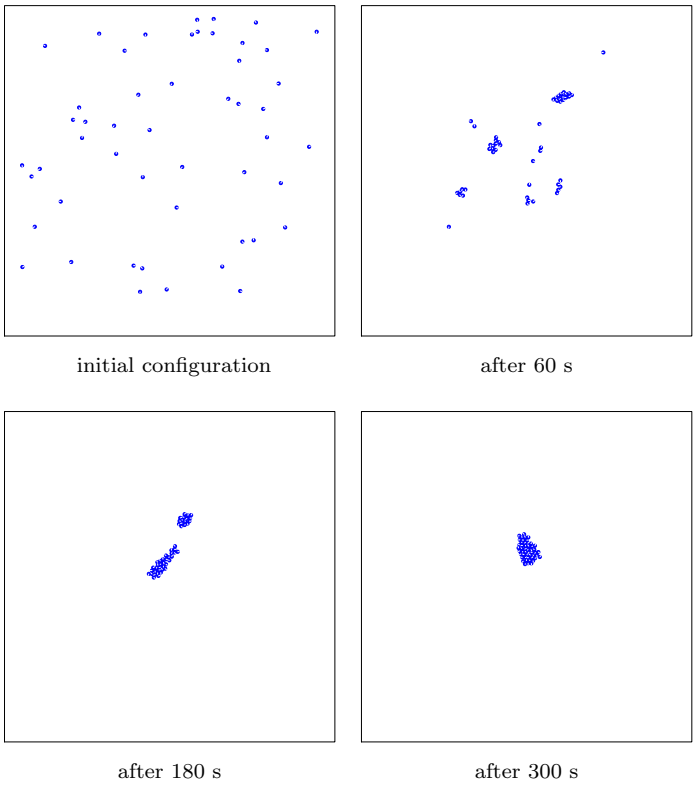


Figure 1.3: Snapshots of the aggregation behavior of 50 agents in simulation.

noise-free (situations with noise being present are considered in Section 1.3.7 and the physical experiments in Chapter **??** where noise is inherently present). We define the linear speed to be positive when the angle between the individual's orientation and its direction of motion is smaller than $\pi/2$ rad, and negative otherwise.

In the following, we detail the behavioral rules of the two swarm behaviors.

1.1.2.2 Aggregation

In this behavior, the sensor is binary, that is, $n = 2$. It gives a reading of $I = 1$ if there is an agent in the line of sight, and $I = 0$ otherwise. The environment is free of obstacles. The objective for the agents is to aggregate into a single compact cluster as fast as possible. Further details, including a validation with 40 physical e-puck robots, are reported in [21].

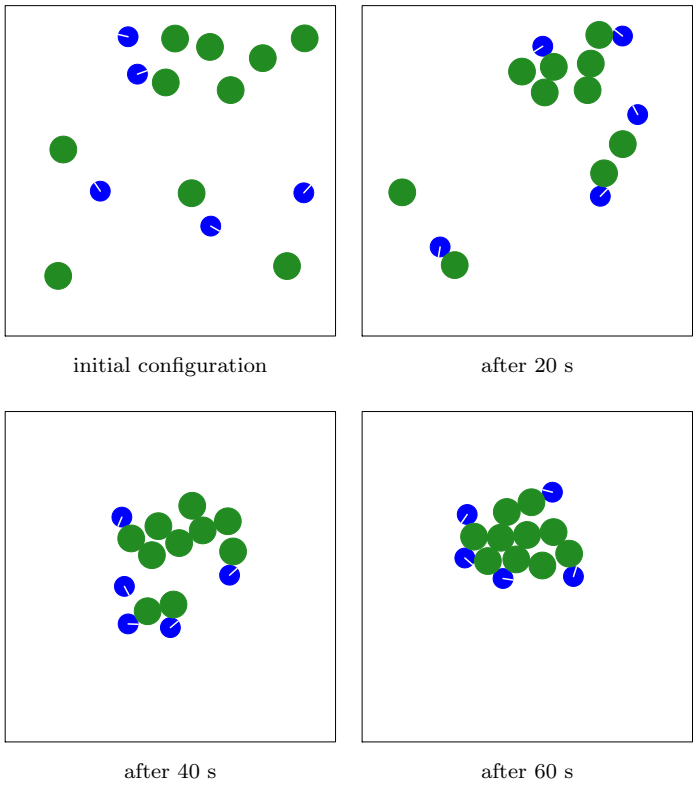


Figure 1.4: Snapshots of the object clustering behavior in simulation. There are 5 agents (blue) and 10 objects (green).

The 'optimal' controller for aggregation was found by performing a grid search over the entire space of possible controllers (with finite resolution) [21]. The 'optimal' controller's parameters are:

$$\mathbf{p} = (-0.7, -1.0, 1.0, -1.0)\,. \tag{1.7}$$

When $I = 0$, an agent moves backwards along a clockwise circular trajectory ($v_{\ell 0} = -0.7$ and $v_{r0} = -1.0$). When $I = 1$, an agent rotates clockwise on the spot with maximum angular velocity ($v_{\ell 1} = 1.0$ and $v_{r1} = -1.0$). Note that rather counter-intuitively, an agent never moves forward, regardless of I. With this controller, an agent provably aggregates with another agent or a quasi-static cluster of agents [21]. Figure 1.3 shows snapshots from a simulation trial with 50 agents.

1.1.2.3 Object Clustering

This behavior uses $n = 3$ sensor states: $I = 0$ if the sensor is pointing at the background (e.g., the wall of the environment, if the latter is bounded), $I = 1$ if the sensor is pointing at an object, and $I = 2$ if it is pointing at another agent. The objective of the agents is to arrange the objects into a single compact cluster as fast as possible. Details of this behavior, including a validation using 5 physical e-puck robots and 20 cylindrical objects, are presented in [22].

The controller's parameters, found using an evolutionary algorithm [22], are:

$$\mathbf{p} = (0.5, 1.0, 1.0, 0.5, 0.1, 0.5)\,. \tag{1.8}$$

When $I = 0$ and $I = 2$, the agent moves forward along an anti-clockwise circular trajectory, but with different linear and angular speeds. When $I = 1$, it moves forward along a clockwise circular trajectory. Figure 1.4 shows snapshots from a simulation trial with 5 agents and 10 objects.

1.2 Simulation Platform and Setups

In this section, we present the agent platform for simulating the two swarm behaviors under investigation and the simulation setups for performing coevolution runs.

1.2.1 Simulation Platform

We use the open-source Enki library [180], which models the kinematics and dynamics of rigid objects, and handles collisions. Enki has a built-in 2-D model of the e-puck. The robot is represented as a disk of diameter 7.0 cm and mass 150 g. The inter-wheel distance is 5.1 cm. The speed of each wheel can be set independently. Enki induces noise on each wheel speed by multiplying the set value by a number in the range $(0.95, 1.05)$ chosen randomly with uniform distribution. The maximum speed of the e-puck is 12.8 cm/s, forward or backward. The line-of-sight sensor is simulated by casting a ray from the

e-puck's front and checking the first item with which it intersects (if any). The range of this sensor is unlimited in simulation.

In the object clustering case study, we model objects as disks of diameter 10 cm with mass 35 g and a coefficient of static friction with the ground of 0.58, which makes it movable by a single e-puck.

The robot's control cycle is updated every 0.1 s, and the physics is updated every 0.01 s.

1.2.2 Simulation Setups

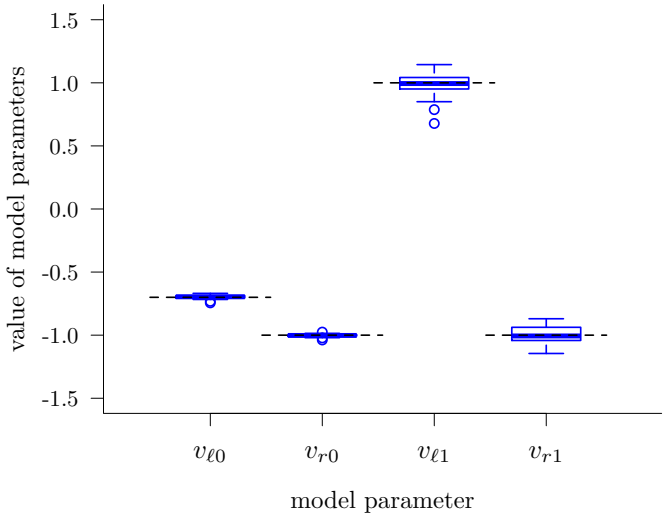
In all simulations, we used an unbounded environment. For the aggregation case study, we used groups of 11 individuals—10 agents and 1 replica that executes a model. The initial positions of individuals were generated randomly in a square region of sides 331.66 cm, following a uniform distribution (average area per individual = 10000 cm^2). For the object clustering case study, we used groups of 5 individuals—4 agents and 1 replica that executes a model—and 10 cylindrical objects. The initial positions of individuals and objects were generated randomly in a square region of sides 100 cm, following a uniform distribution (average area per object = 1000 cm^2). In both case studies, individual starting orientations were chosen randomly in $[-\pi, \pi]$ with uniform distribution.

We performed 30 coevolution runs for each case study. Each run lasted 1000 generations. The model and classifier populations each consisted of 100 solutions ($\mu = 50$, $\lambda = 50$). In each trial, classifiers observed individuals for 10 s at 0.1 s intervals (100 data points).

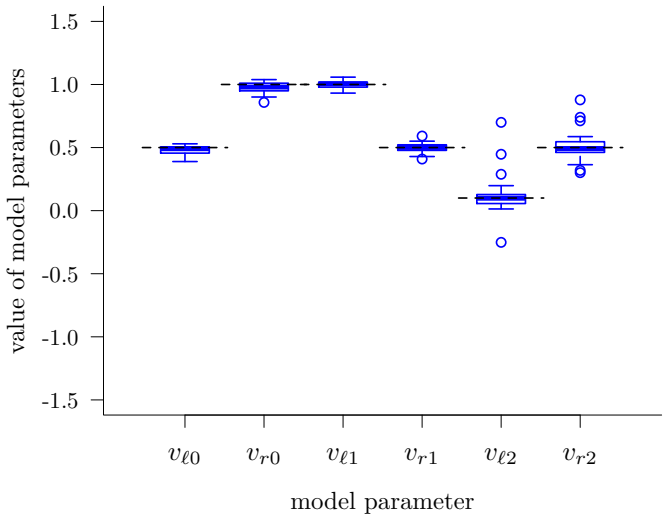
1.3 Simulation Results

1.3.1 Analysis of Evolved Models

In order to objectively measure the quality of the models obtained through *Turing Learning*, we define two metrics. Given an evolved controller (model) $\mathbf{x}$ and the original agent



(a) Aggregation



(b) Object Clustering

Figure 1.5: Parameters of the evolved models with the highest subjective fitness in the 1000^{th} generation in the coevolutions for (a) the aggregation behavior and (b) the object clustering behavior. Each box corresponds to 30 coevolution runs in simulation. The dotted black lines correspond to the values of the parameters that the system is expected to learn (i.e., those of the agent).

controller $\mathbf{p}$, where $\mathbf{x}, \mathbf{p} \in [-1, 1]^{2n}$, we define the absolute error (AE) in a particular parameter $i \in \{1, 2, \ldots, 2n\}$ as:

$$\text{AE}_i = |x_i - p_i|. \tag{1.9}$$

We define the mean absolute error (MAE) over all parameters as:

$$\text{MAE} = \frac{1}{2n} \sum_{i=1}^{2n} \text{AE}_i. \tag{1.10}$$

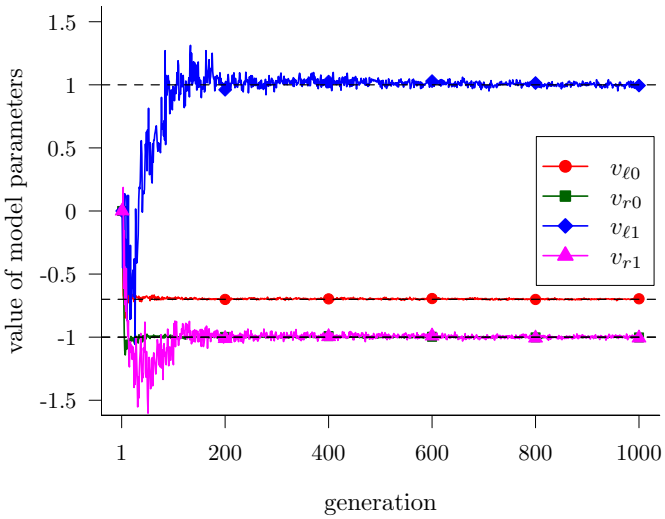
Fig. 1.5 shows a box plot[5] with the parameters of the evolved models with the highest subjective fitness[6] in the final generation. It can be seen that *Turing Learning* identified the parameters for both behaviors with good accuracy (dotted black lines represent the ground truth, that is, the parameters of the observed swarming agents). In the case of aggregation, the means (standard deviations) of the AEs in the parameters were (from left to right in Fig. 1.5(a)): 0.01 (0.01), 0.01 (0.01), 0.07 (0.07) and 0.06 (0.04). In the case of object clustering, these values were: 0.03 (0.03), 0.04 (0.03), 0.02 (0.02), 0.03 (0.03), 0.08 (0.13) and 0.08 (0.09).

We also investigated the evolutionary dynamics. Fig. 1.6 shows how the model parameters converged over generations. In the aggregation case study, the parameters corresponding to $I = 0$ were learned first. After around 50 generations, both $v_{\ell 0}$ and v_{r0} closely approximated their true values (-0.7 and -1.0), shown in Fig. 1.6(a). For $I = 1$, it took about 200 generations for both $v_{\ell 1}$ and v_{r1} to converge. A likely reason for this effect is that an agent spends a larger proportion of its time seeing nothing ($I = 0$) than other agents ($I = 1$)—simulations revealed these percentages to be 91.2% and 8.8%, respectively (mean values across 100 trials).

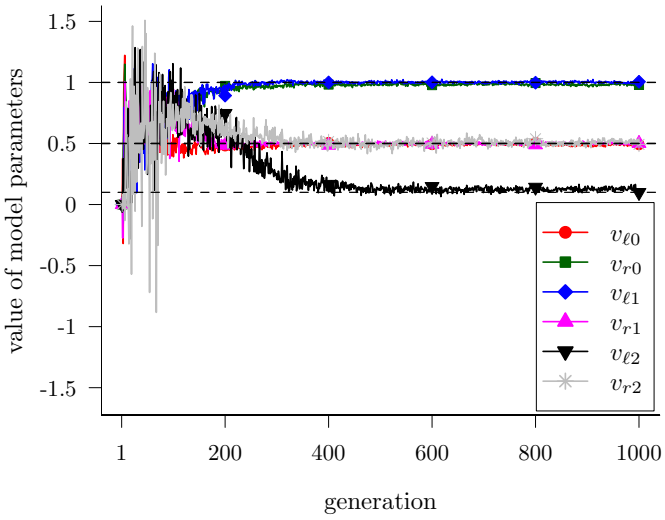
In the object clustering case study, the parameters corresponding to $I = 0$ and $I = 1$ were learned faster than the parameters corresponding to $I = 2$, as shown in Fig. 1.6(b). After about 200 generations, $v_{\ell 0}$, v_{r0}, $v_{\ell 1}$ and v_{r1} started to converge; however it took about 400 generations for $v_{\ell 2}$ and v_{r2} to approximate their true values. Note that an agent spends the highest proportion of its time seeing nothing ($I = 0$), followed by objects ($I = 1$) and other agents ($I = 2$)—simulations revealed these proportions to be 53.2%, 34.2% and 12.6%, respectively (mean values across 100 trials).

Although the evolved models approximate the agents well in terms of parameters, it

[5] The box plots presented here are all as follows. The line inside the box represents the median of the data. The edges of the box represent the lower and the upper quartiles of the data, whereas the whiskers represent the lowest and the highest data points that are within 1.5 times the range from



(a) Aggregation



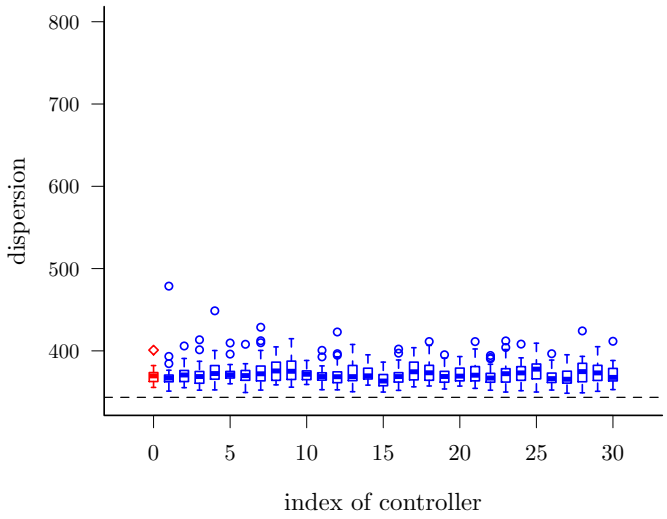
(b) Object Clustering

Figure 1.6: Evolutionary process of the evolved model parameters for (a) the aggregation behavior and (b) the object clustering behavior. Curves represent median values across 30 coevolution runs. Dotted black lines indicate true values.

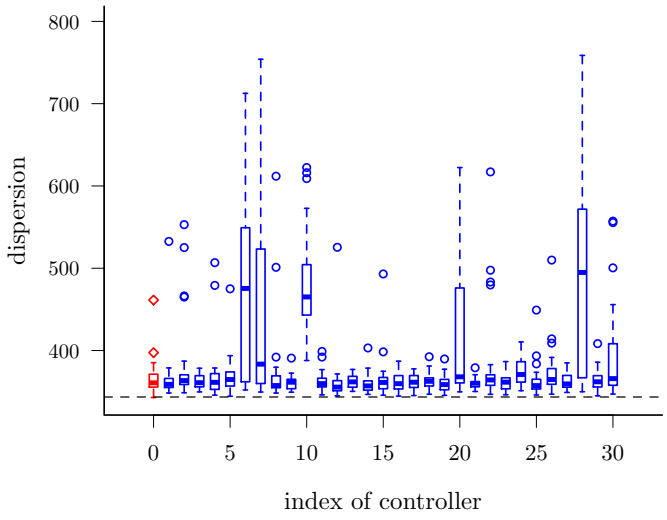
has often been observed in swarm systems that small changes in individual agent behaviors can lead to vastly different emergent behaviors, especially with large numbers of agents [182]. For this reason, we evaluated the quality of the emergent behaviors that the models give rise to. In the case of aggregation, a good measure of the emergent behavior

the lower and the upper quartiles, respectively. Circles represent outliers.

[6]The fitness of the models depends solely on the judgments of the classifiers from the competing population, and is hence referred to as *subjective*.



(a) Aggregation



(b) Object Clustering

Figure 1.7: (a) Dispersion (after 400 s) of 50 agents executing the original aggregation controller (red box) or one of the 30 evolved models (blue boxes) of the 1000$^{\text{th}}$ generation. (b) Dispersion (after 400 s) of 50 objects in a swarm of 25 agents executing the original object clustering controller (red box) or one of the 30 evolved models (blue boxes). In both (a) and (b), boxes show distributions over 30 trials. The dotted black lines indicate the minimum dispersion that 50 agents/objects can possible achieve [181]. See Section 1.3.1 for details.

is the dispersion of the swarm after some elapsed time as defined in [21][7]. For each of the 30 models with the highest subjective fitness in the final generation, we performed 30

[7]The measure of dispersion is based on the robots'/objects' distances from their centroid. For a formal definition, see Equation (5) of [21], Equation (2) of [22] and [181].

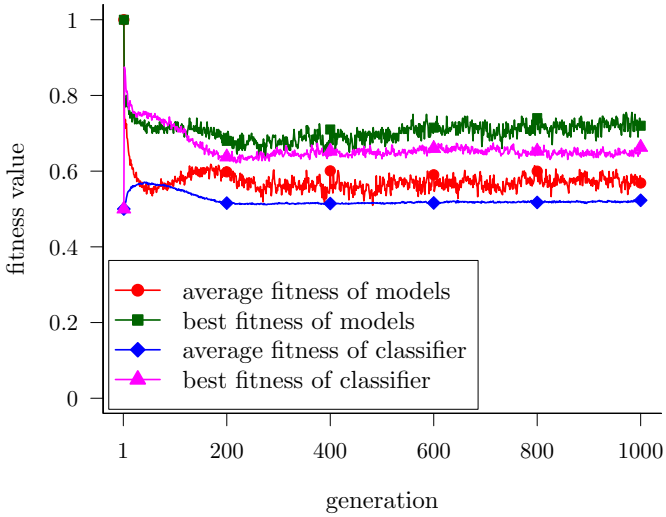
trials with 50 agents each executing the model. For comparison, we also performed 30 trials using the original controller (see Equation (1.7)). The set of initial configurations was the same for all models and the original controller. Fig 1.7(a) shows the dispersion for the original controller and models after 400 s. All models led to aggregation. We performed a statistical test[8] on the final dispersion of the agents between the original controller and each model. There was no statistically significant difference in 26 out of 30 cases (30 out of 30 cases with Bonferroni correction).

In the case of object clustering, we use the dispersion of the objects after some elapsed time as a measure of the emergent behavior. With the original controller (see Equation (1.8)) and each of the models, we performed 30 trials with 25 agents and 50 objects. The results are shown in Fig. 1.7(b). In a statistical test on the final dispersion of the objects between the original controller and each model, there was no statistically significant difference in 24 out of 30 cases (26 out of 30 cases with Bonferroni correction).

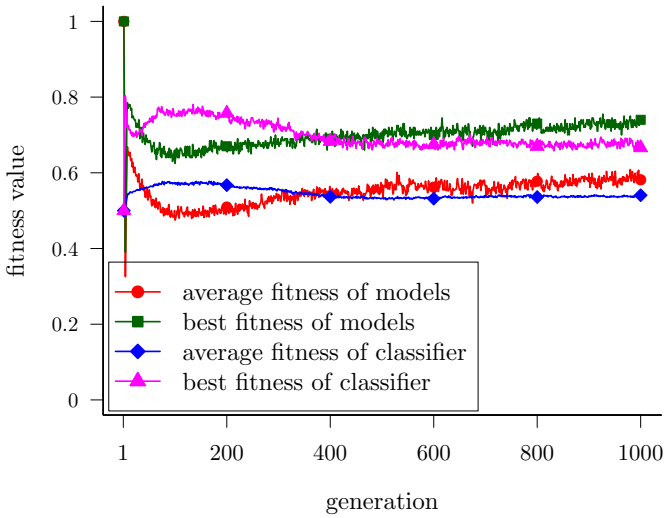
We also investigated the evolutionary process of the model parameters. Figure 1.6 shows the convergence of the model parameters over generations. In the aggregation behavior, the parameters corresponding to $I = 0$ were learned first. After about 50 generations, both $v_{\ell 0}$ and v_{r0} closely approximated their true values (-0.7 and -1.0) shown in Figure 1.5(a). For $I = 1$, it took about 200 generations for both v_{l1} and v_{r1} to converge. A likely reason for this effect is that an agent spends a larger percentage of its time seeing nothing ($I = 0$) than other agents ($I = 1$)—simulations revealed these percentages to be 8.8% and 91.2%, respectively (mean values over 100 trials).

In the object clustering behavior, the parameters corresponding to $I = 0$ and $I = 1$ were learned faster than the other two parameters corresponding to $I = 2$, as shown in the Figure 1.6(b). After about 200 generations, $v_{\ell 0}$, v_{r0}, v_{l1} and v_{r1} started to converge; however it took about 400 generations for v_{l2} and v_{r2} to approximate their true values. This is likely because an agent spends the most percentage of its time seeing nothing ($I = 0$), followed by objects ($I = 1$) and other agents ($I = 2$)—simulations revealed these percentages to be 53.2%, 34.2% and 12.6% , respectively (mean values over 100 trials).

[8] Throughout this thesis, the statistical test used is a two-sided Mann-Whitney test with a 5% significance level.



(a) Aggregation



(b) Object Clustering

Figure 1.8: This plot shows the subjective fitness of the classifiers and the models for (a) the aggregation behavior and (b) the object clustering behavior. The curves show the median value across 30 coevolution runs.

1.3.2 Coevolutionary Dynamics

In order to analyze how the classifiers and the models interact with each other during the course of the coevolution, we investigate the dynamics of the subjective fitness of the classifiers and the models as shown in Figure 1.8.

In the aggregation behavior, at the beginning, the fitness of the classifiers is 0.5 as they

output 1 for all the agents and models, which means the classifiers make uninformed decisions[9]. Therefore, the fitness of the models starts from 1.0, as all the classifiers judge them the agent. Then, the average fitness of the classifiers quickly increases, corresponding to the decline of the average fitness of the models. As the models learn to adapt, the average fitness of the classifiers only increases slightly until about the 50^{th} generation. After that, the average fitness of the models starts to increase. However, the best fitness of the classifiers is still higher than that of the models. The best fitness of the models surpasses that of the classifiers after about the 120^{th} generation; at this point, the fitness of the 'best' model (selected by the classifiers) is around 0.7. This means that the 'best' model is able to mislead 70% of the classifiers into judging it as the agent. From the 200^{th} generation onwards, the fitness of the classifiers and models remains "balanced" until the last generation.

The coevolutionary dynamics of the object clustering behavior is similar. Compared with the dynamics of the aggregation behavior, it takes more generations for the fitness of the models to surpass that of the classifiers. This could be explained by the higher number of parameters and the higher complexity of the behavior to be evolved in the object clustering behavior.

1.3.3 Analysis of Evolved Classifiers

The primary outcome of the *Turing Learning* method (and of any system identification method) is the model, which has been discussed in the previous section. However, the evolved classifiers can also be considered as a useful byproduct. For instance they could be used to detect abnormal agents in a swarm. We will now analyze the performance of the evolved classifiers. For the remainder of this chapter, we consider only the aggregation case study.

To assess the performance of the classifiers, we measure the percentage of correct judgments over agents and a wide range of models. The models are uniformly distributed

[9]Note that in the implementation of the coevolutionary algorithm, the parameters of the models and classifiers are initialized to 0.0, which means all the classifiers are identical at the 1^{st} generation.

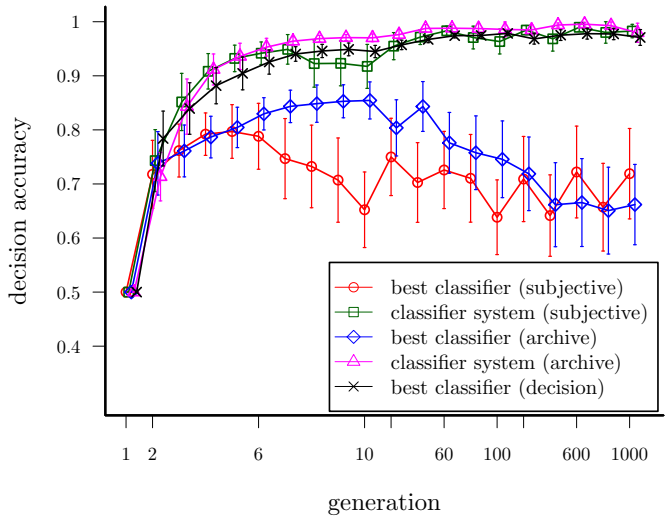


Figure 1.9: The average decision accuracy of the best classifiers and classifier systems over generations (nonlinear scale) in 30 coevolution runs. The error bars show standard deviations. See text for details.

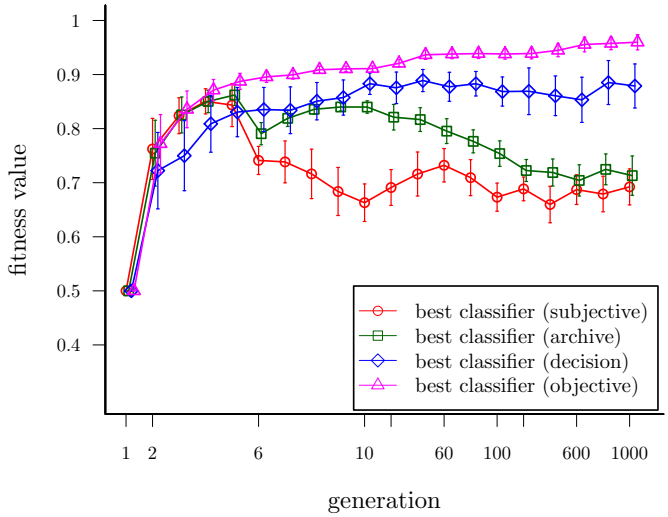


Figure 1.10: This plot shows the average fitness of the classifiers over generations (nonlinear scale) in 30 coevolution runs. The fitness is calculated based on 14641 testing models. See text for details.

across the entire parameter space of the agents: $[-1, 1]^4$. To keep the analysis of classifiers within a reasonable computation time, we discretize this space using 11 settings per parameter, to obtain: $\mathcal{X} = \{-1.0, -0.8, ..., 0.8, 1.0\}^4$. This discretized space is a grid consisting of $|\mathcal{X}| = 11^4 = 14641$ points (i.e., models). The classifier's performance is computed as follows. The model is executed by a replica mixed into a group of 10 agents

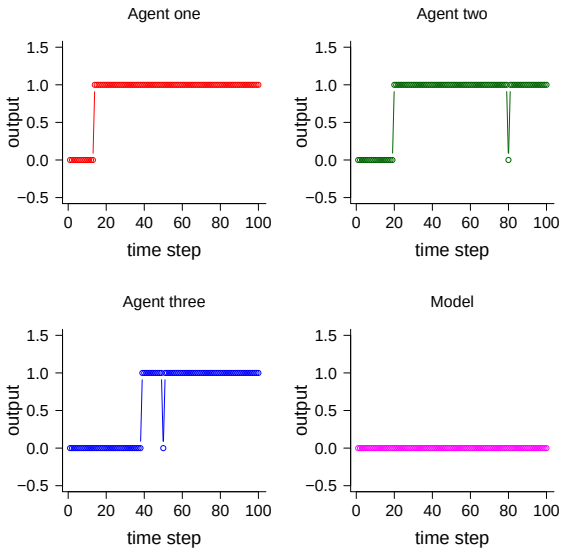
(as in the coevolution runs). 10 trials are performed using a set of initial configurations common to all classifiers. The motion data is fed to each classifier, which makes 10 judgments per individual. If the classifier consistently judges the individual as a model (i.e. not an agent) in 10 out of 10 trials, it outputs a "model" decision. Otherwise, it outputs "agent". This conservative approach was used to minimize the risk of false positive detection of abnormal behavior.

1.3.3.1 Using a Single Classifier

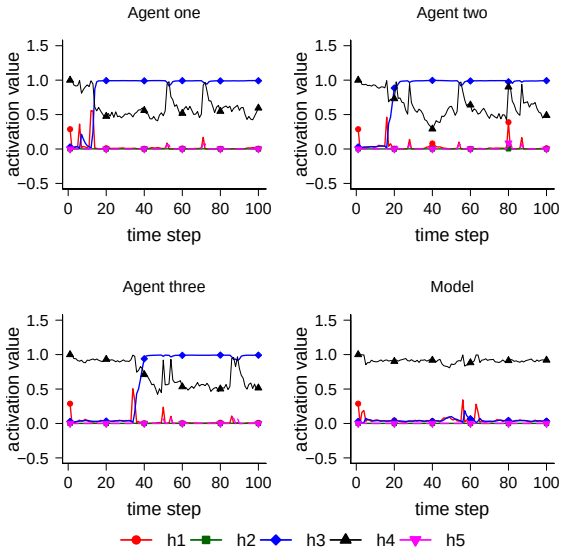
The average decision accuracy of the classifier with the highest subjective fitness in 30 coevolution runs is shown in Fig. 1.9 (*best classifier (subjective)*). The accuracy combines the percentage of correct judgments about models (50% weight) with the percentage of correct judgments about agents (50% weight). The accuracy of the classifier increases in the first 5 generations, then drops and fluctuates within range 62%–80%.

An alternative strategy is to select the classifier that achieves the highest fitness when evaluated on the whole historical tracking data (not just those of the current generation). The decision accuracy of this classifier is also shown in Fig. 1.9 (*best classifier (archive)*). The trend is similar to that of *best classifier (subjective)*. The accuracy increases in the first 10 generations, and then starts *decaying*, dropping to around 65% by the 1000^{th} generation. However, in the earlier generations, the accuracy of the *best classifier (archive)* is higher than that of the *best classifier (subjective)*. For a comparison, we also plot the highest decision accuracy that a single classifier achieves for each generation (*best classifier (decision)*). Interestingly, the accuracy of the *best classifier (decision)*, which is shown in Fig. 1.9 (black curve), increases almost monotonically, reaching a level above 95%. Note that to select the *best classifier (decision)*, one needs to perform additional trials (146410 in this case).

At first sight, it is counter-intuitive that selecting the best classifier according to the historical data still leads to low decision accuracy. This phenomenon, however, can be explained when considering the model population. We have shown in the previous section (see especially Fig. 1.6(a)) that the models converge rapidly at the beginning of the coevolutions. As a result, when classifiers are evaluated in later generations, the trials are likely to include models very similar to each other. Classifiers that become



(a) decision-making process



(b) activation of hidden neurons

Figure 1.11: This plot shows the (a) decision-making process and (b) the corresponding activation value of the 5 hidden neurons of a classifier for three random-chosen agents and the replica that executes a very good model in a trial. Hidden neurons are labeled as h1, h2, h3, h4, and h5.

overspecialized to this small set of models (the ones dominating the later generations) have a higher chance of being selected in the post-evaluation. These classifiers may however have a low performance when evaluated across the entire model space.

To investigate whether the decision accuracy is a good measurement of the quality of the classifiers, we plot a figure showing the fitness of the best classifiers corresponding to Figure 1.9. The fitness is calculated based on the 14641 models. In addition, we also plot the classifier with the highest fitness for comparison, which is termed as *best classifier (objective)*. The results are shown in Figure 1.10. As we can see, the trend of fitness curves show good correspondence to that of the decision accuracy in Figure 1.9, although there is still a small gap between the *best classifier (decision)* and *best classifier (objective)*. This means measuring the quality of classifiers according to decision accuracy is a reasonable choice.

In order to understand how the classifiers make judgment, we analyze the internal processing through monitoring the activation of hidden neurons. Figure 1.11 shows the decision-making process and the corresponding activation value of the 5 hidden neurons of the classifier with the highest decision accuracy in the last generation of a coevolution run for 3 randomly-chosen agents and the replica in a trial. The model executed on the replica has a parameter set of $(-0.7, -1.0, 1.0, -0.9)$, which is very near to that of the agent in Equation (1.7). As we can see in Figure 1.11(a), for the agents, the classifier outputs 0 at the beginning, and after a certain time period outputs 1. This means the classifier needs some time to make the correct judgment. Note that for some time steps after it started to output 1, it still outputs 0, but this happens only occasionally. For the model, it always outputs 0. This phenomenon can be explained by the activation value of the hidden neurons of the classifier. The classifier's activation value of the hidden neurons (labeled as h1, h2, h3, h4, and h5 in Figure 1.11(b)) reveals how it processes the motion of different individuals[10]. It seems that the decision-making ("sudden jump" in Figure 1.11(a)) of the classifier is related with the variation of h3 and h4. Consider the classification of the agents for example, at the beginning, h3 has a very low value (which is almost 0) and then after some time it starts to increase. The variation tendency of h4 is opposite. Every time the classifier's judgment starts to jump from 0 to 1, the value of h3 starts to surpass that of h4. h1, h2 and h5 are almost not activated during the whole period, although sometimes the activation of h2 influences the judgment of the classifier. For instance, there are two peaks at about 50^{th} and 80^{th} time step in h2 for agent two and three, which correspond to a wrong judgment of the classifier (that is,

[10]Note that the activation of hidden neurons may not be the same for all the classifiers due to the symmetry of neural networks, but the dynamics is similar.

it outputs 0) at the same point. For the model, all the hidden neurons are almost not activated.

1.3.3.2 Using a Classifier System

The results of the previous section have shown that using a single classifier is not a good solution; although there may be a good classifier in each generation, it may take significant effort (trials with agents) to find it.

To address this problem, we propose the use of a classifier system, that is, a number of classifiers working in tandem to judge a given candidate. We choose the best 10 classifiers[11]. This is either the set of 10 classifiers with the highest subjective fitness in the current generation or the set of classifiers that achieve the highest fitness when evaluated on the whole historical data. If one or more classifiers make a decision about the candidate as a model (i.e. not an agent), the system outputs a "model" decision. Otherwise, it outputs "agent".

The results of using a classifier system are shown in Fig. 1.9 (green and magenta, respectively). The two systems exhibit significantly improved decision accuracy across all generations. After 1000 generations, each system has a high accuracy of above 95%, on average.

When post-evaluating the performance of classifier systems, we kept the setup the same as the one used in the coevolution runs (i.e., 10 agents and 1 replica in each trial). In practice, the ratio of agents and replicas in a trial could be different. Therefore, we analyzed the scalability of the classifier systems selected in the last generation over 30 coevolution runs with different number of agents in the group. Note that there is always one replica in the group. We tested 14641 different models. Figure 1.12 shows the

[11] Note that the number of classifiers chosen to form the system here is not necessarily optimal. There is no guarantee that the individually best few classifiers will form the best *system* when working in tandem. In principle, one could exhaustively search every possible combination of a given number of classifiers from the population. However, this is often infeasible—with our settings of choosing 10 classifiers out of 100, 1.73×10^{13} possibilities exist. We therefore propose the heuristic of choosing the individually best classifiers to form the system, and empirically show that this nevertheless yields good results.

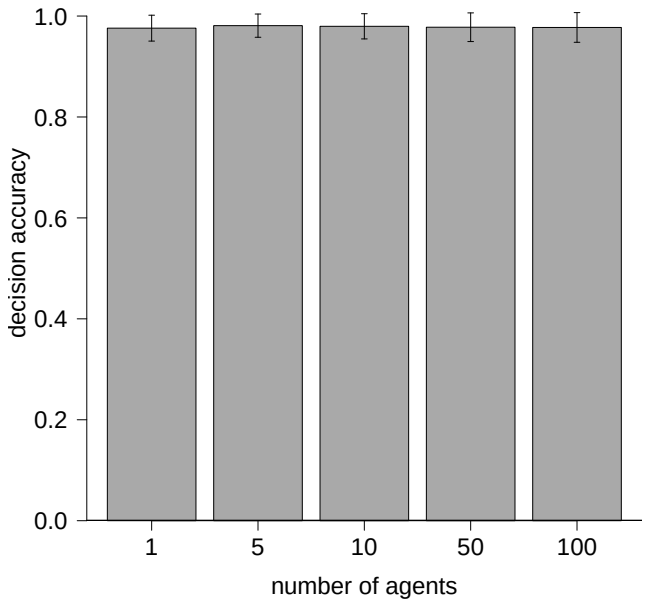


Figure 1.12: This plot shows the decision accuracy of the *classifier system (archive)* in the 1000^{th} generation of 30 coevolution runs for different numbers of agents in the group. A single replica was present in each trial. In each case, 14641 different models were tested.

decision accuracy of the *classifier system (archive)* when changing the number of agents in the group. As we can see, the decision accuracy of the classifier system is not affected by the variation of the number of agents, which shows the robustness of the classifier system.

As we mentioned before, we chose 10 classifiers to form a classifier system to have a reasonable decision accuracy. Here we investigate how the classifier systems perform with various number of classifiers chosen to form a system. The decision accuracy of the *classifier system (archive)* is shown in Figure 1.13[12]. It seems that as long as the number of classifiers chosen to form a system is within a certain range, the system could perform well. For example, when the number of selected classifiers is between 10 and 25, the classifier system can obtain a high decision accuracy. However, when the number is high (e.g., 75), the decision accuracy declines dramatically.

[12]The result of *classifier system (subjective)* is similar.

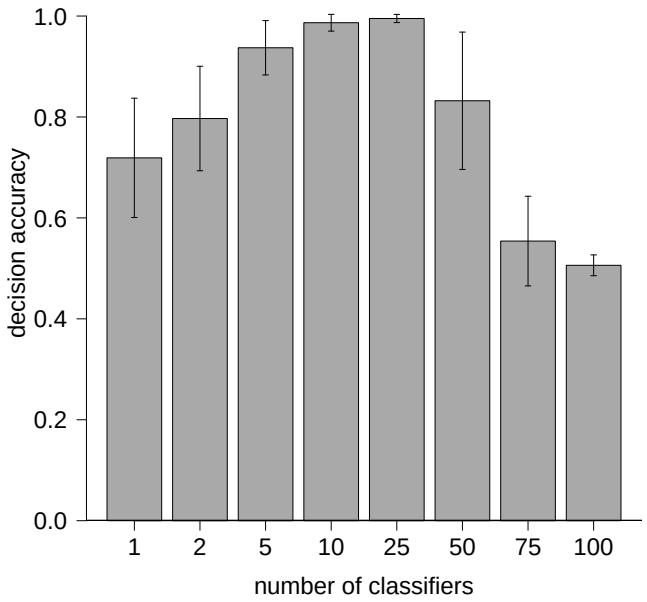


Figure 1.13: This plot shows the decision accuracy of the *classifier system (archive)* in the 1000$^{\text{th}}$ generation of 30 coevolution runs with various number of classifiers chosen to form a system.

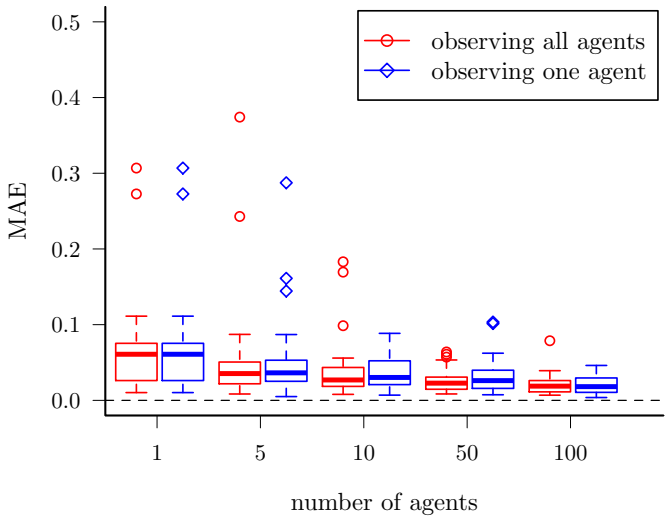


Figure 1.14: MAE (defined in Equation (1.10)) of the evolved models with the highest subjective fitness after 1000 generations, when using *Turing Learning* with varying numbers of agents (excluding the replica). Red and blue boxes show, respectively, the cases where all agents are observed, and one agent is observed. Boxes show distributions over 30 coevolution runs.

1.3.4 Observing Only a Subset of Agents

So far, we have used motion data about all agents in the swarm when evaluating classifiers. However, this may not always be feasible in practice. For instance, given a video recording of a large and/or dense swarm, extracting motion data about all agents may be infeasible or lead to highly inaccurate results. A more practical solution might be to only track a subset of agents (e.g., by equipping them with markers).

We now compare the case where all agents are observed with the other extreme, where only one agent is observed. We study how these two cases compare as the swarm size increases. We conducted 30 coevolution runs with each number of agents $n \in \{1, 5, 10, 50, 100\}$. There was always one replica in the group. When observing only one agent, this was chosen randomly in each trial. Note that the total number of trials in each coevolution run for the case of observing all agents and one agent is identical. We measured the total square error of the model with the highest subjective fitness in the last (1000^{th}) generation of each run. The results are shown in Fig. 1.14.

There is no statistically significant difference for any n. On the other hand, as the swarm size increases, performance improves. For example, there is a statistically significant difference between $n = 10$ and $n = 100$, both when observing all agents and one. These results suggest that the key factor in the coevolutionary process is not the number of observed agents, so much as the richness of information that comes from increasing inter-agent interactions, and is reflected in each agent's motion. This means that in practice, observing a single agent is sufficient to infer accurate models, as long as the swarm size is sufficiently large.

A similar phenomenon was observed in a different scenario, where the goal was to distinguish between different 'modes' of a swarm (i.e., global behaviors) through observing only a few individuals [183].

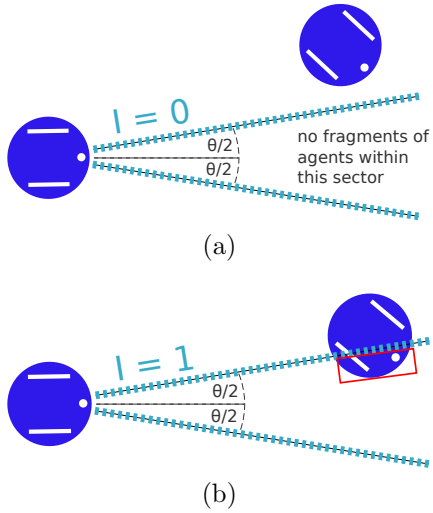


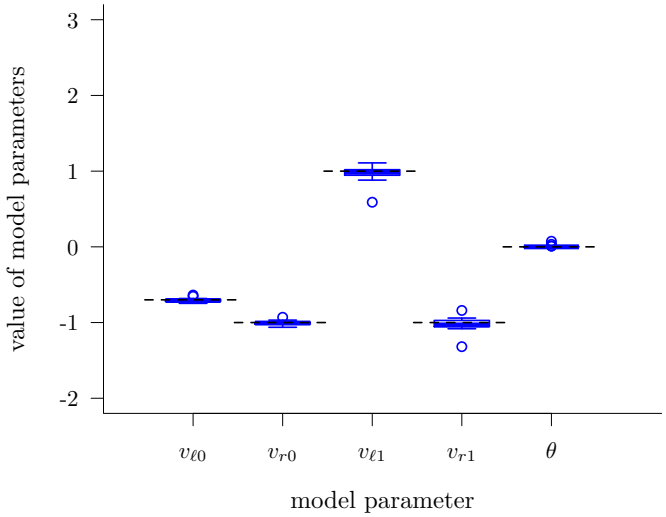
Figure 1.15: A diagram showing the angle of view of the agent's sensor investigated in Section 1.3.5.

1.3.5 Evolving Control and Morphology

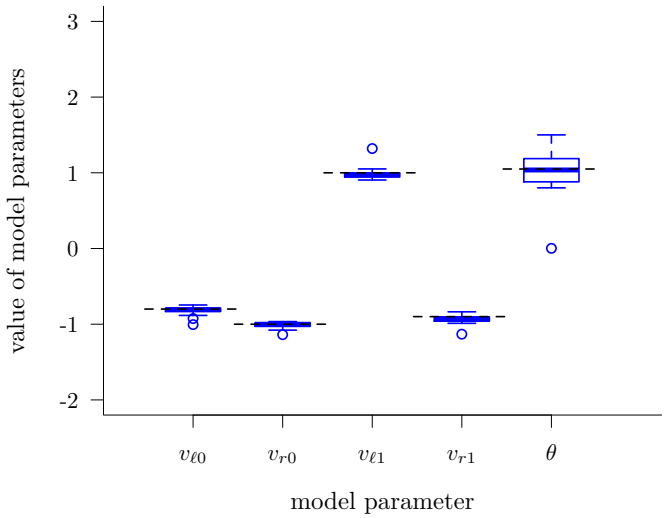
In the previous sections, we assumed that we fully knew the agents' morphology (i.e., structure), and only their behavior (controller) was to be identified. We now present a variation where one aspect of the morphology is also unknown. The replica, in addition to the four controller parameters, takes a parameter $\theta \in [0, 2\pi]$ rad, which determines the horizontal field of view of its sensor, as shown in Fig. 1.15 (however, the sensor is still binary). Note that in the previous sections the agents' line-of-sight sensor can be considered as a sensor with angle of view of 0 rad.

The models now have five parameters. As before, we let the coevolution run in an unbounded search space (i.e., now, $\mathbb{R}^5$). However, as θ is necessarily bounded, before a model was executed on a replica, the parameter corresponding to θ was mapped to the range $[0, 2\pi]$ using a logistic sigmoid function (Equation (1.1)). The controller parameters were directly passed to the replica. In this setup, the classifiers observed the individuals for 100 s in each trial (preliminary results indicated that this setup requires a longer observation time).

Fig. 1.16(a) shows the parameters of the subjectively best models in the last (1000^{th}) generations of 30 coevolution runs. The means (standard deviations) of the AEs in each model parameter were: 0.02 (0.01), 0.02 (0.02), 0.05 (0.07), 0.06 (0.06) and 0.01 (0.01).



(a)

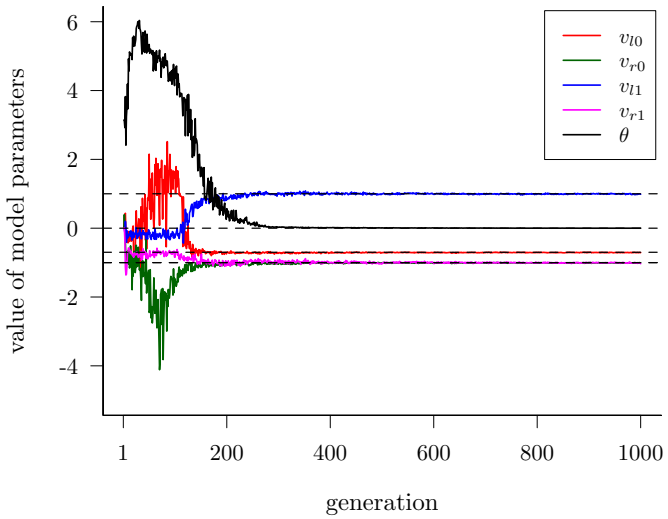


(b)

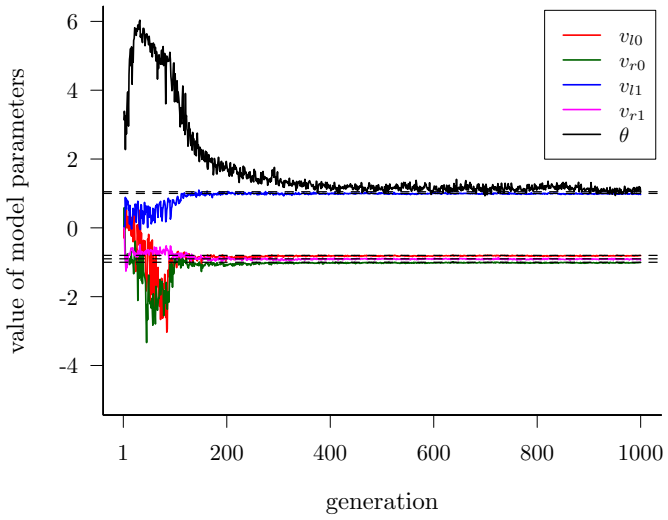
Figure 1.16: Parameters (controller parameters and angle of view in rad) of the evolved models with the highest subjective fitness in the 1000$^{\text{th}}$ generation corresponding to the case of the agents' angle of view equal to (a) 0 rad and (b) $\pi/3$ rad. Boxes show distributions over 30 coevolution runs. Dotted black lines indicate true values.

All parameters including θ were still learned with high accuracy.

The case where the true value of θ is 0 rad is an edge case, because given an arbitrarily small $\epsilon > 0$, the logistic sigmoid function maps an infinite domain of values onto $(0, \epsilon]$. This makes it easier for the coevolution to learn this parameter. For this reason, we also considered another scenario where the agents' angle of view is $\pi/3$ rad rather than 0 rad.



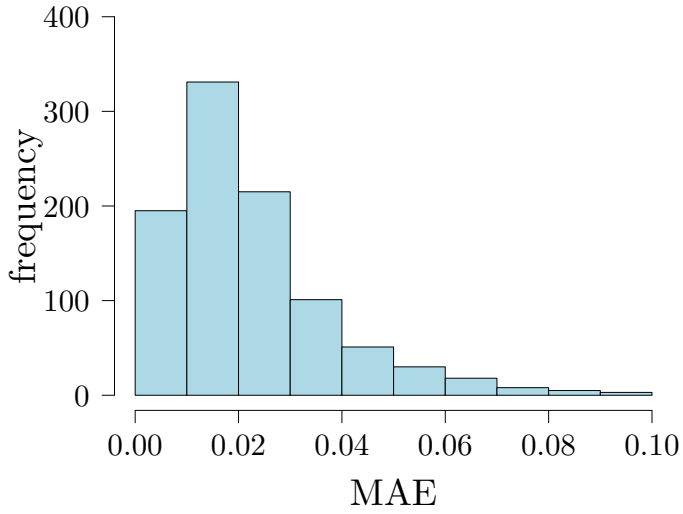
(a)



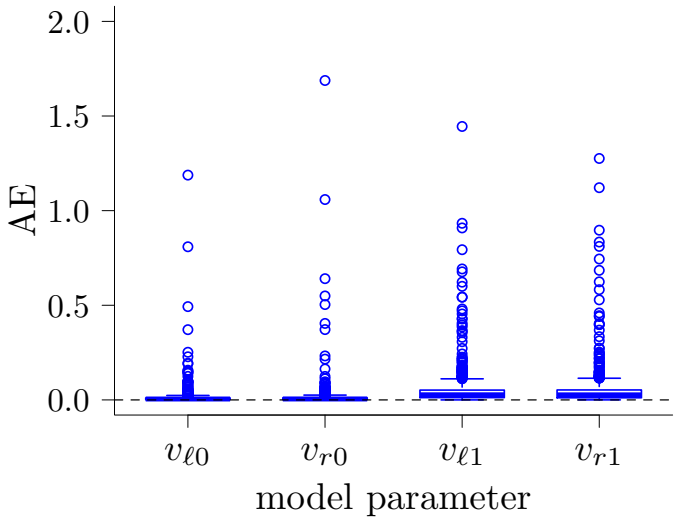
(b)

Figure 1.17: Evolutionary process of the evolved models (controller parameters and angle of view in rad) in the 1000$^{\text{th}}$ generation corresponding to the case of the agents' angle of view equal to (a) 0 rad and (b) $\pi/3$ rad. Boxes show distributions over 30 coevolution runs. Dotted black lines indicate true values..

The controller parameters for achieving aggregation in this case are different from those in Equation (1.7). They were found by re-running a grid search with the modified sensor. Fig. 1.16(b) shows the results from 30 coevolutions with this setup. The means (standard deviations) of the AEs in each parameter were: 0.04 (0.04), 0.03 (0.03), 0.05 (0.06), 0.05 (0.05) and 0.20 (0.19). The controller parameters were still learned with good accuracy.



(a)



(b)

Figure 1.18: This plot shows (a) a histogram of the MAE (defined in Equation (1.10)) of the evolved models and (b) the AEs (defined in Equation (1.9)) of each model parameter in the 1000^{th} generation over 1000 random behaviors. For each behavior, we performed one coevolution run. In (a), 43 points that have MAE larger than 0.1 are not shown.

The accuracy in the angle of view is noticeably lower, but still reasonable.

1.3.6 Evolving Other Behaviors

The aggregation controller that agents used in our case study was originally synthesized by searching over the space $[-1,1]^4$, using a metric to assess the swarm's global performance [21]. Other points in this space lead to different global behaviors that can be 'meaningful' to a human observer (e.g. circle formation [139]).

We now investigate whether our coevolutionary method can learn arbitrary controllers in this space, irrespective of the global behaviors they lead to. We generated 1000 controllers randomly in $[-1,1]^4$, with uniform distribution. For each controller we performed one coevolution run, and selected the subjectively best model in the last (1000^{th}) generation.

Fig. 1.18(a) shows a histogram of the MAE of the evolved models. The distribution has a single mode close to zero, and decays rapidly for increasing values. Over 89% of the 1000 cases have an error below 0.05. This suggests that the accuracy of *Turing Learning* is not highly sensitive to the particular behavior under investigation (i.e., most behaviors are learned equally well). Fig. 1.18(b) shows the AEs of each model parameter. The means (standard deviations) of the AEs in each parameter were: 0.01 (0.05), 0.02 (0.07), 0.07 (0.6) and 0.05 (0.20). We performed a statistical test on the AEs between the model parameters corresponding to $I = 0$ ($v_{\ell 0}$ and v_{r0}) and $I = 1$ ($v_{\ell 1}$ and v_{r1}). The AEs of the evolved $v_{\ell 0}$ and v_{r0} were significantly lower than those of the evolved $v_{\ell 1}$ and v_{r1}. This was likely due to the reason reported in Section 1.3.1; that is, an agent spends more time seeing nothing ($I = 0$) than other agents ($I = 1$) in each trial.

1.3.7 Noise Study

We conducted a study to investigate how the performance of *Turing Learning* is affected by noise on the measurements of the individuals' position and orientation. As the e-puck robot has a maximum linear speed of 12.8 cm/s, the maximum distance that it can travel in one control cycle (0.1 s) is 1.28 cm. For this reason, we define a disturbance of 1.28 cm to an individual's position as a measurement error of 100%. Similarly, we define

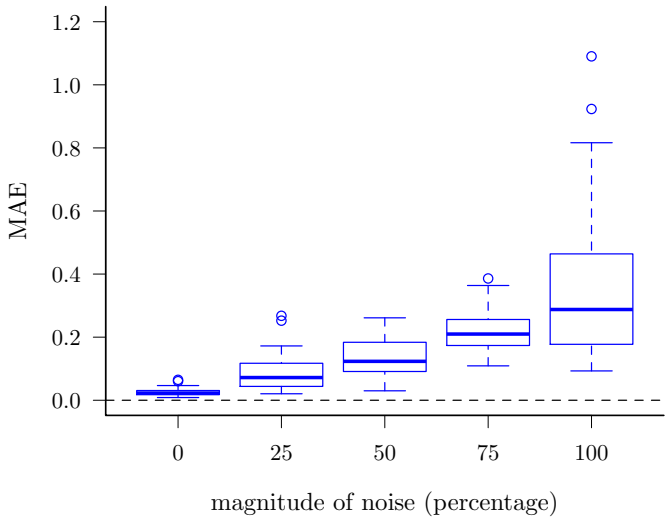


Figure 1.19: This plot shows the MAE (defined in Equation (1.10)) of the evolved parameters when increasing the percentage of noise on the measurements of the individuals' position and orientation in the aggregation behavior. Each box corresponds to 30 coevolution runs. See text for details.

a 100% measurement error on the individual's orientation as the maximum change in orientation that an e-puck can undergo in one control cycle. This corresponds to 0.5 rad.

We conducted 5 sets of 30 coevolutions for the noise values $M = \{0, 25, 50, 75, 100\}\%$. In a coevolution with a noise value M, every measurement of an individual's position is perturbed in a random direction and by a random distance chosen uniformly in $\left[0, 1.28\frac{M}{100}\right]$ cm. Similarly, every orientation measurement is perturbed by a random angle chosen uniformly in $\left[-0.5\frac{M}{100}, 0.5\frac{M}{100}\right]$.

Figure 1.19 shows MAE (defined in Equation (1.10)) of the evolved parameters in the final generations of the coevolutions. This plot reveals that the system still performs relatively well when a considerable amount of noise affects the individuals' motion tracking system.

1.4 Summary

This chapter has presented the simulation results of using the *Turing Learning* method to autonomously learn swarm behaviors through observation. To our knowledge, *Turing*

Learning is the first system identification method that does not rely on any predefined metric to quantitatively gauge the difference between agents and learned models. This eliminates the need to choose a suitable metric and the bias that such metric may have on the obtained solutions.

Through competitive coevolution of models and classifiers, the system successfully learned two swarm behaviors (self-organized aggregation and object clustering). Both the model parameters, which were automatically inferred, and emergent global behaviors closely matched those of the original swarm system. We also constructed a robust classifier system that, given an individual's motion data, can tell whether the individual is an original agent or not. Such classifier system could be effective in detecting abnormal behavior, for example, when faults occur in some members of the swarm. Note that *Turing Learning* produces these classifiers automatically without the need to define a priori what constitutes abnormal behavior.

A scalability study showed that the interactions in a swarm can be characterized by the effects on a subset of agents. In other words, when learning swarm behaviors especially with large number of agents, instead of considering the motion of all the agents in the group, we could focus on a subset of agents. This becomes critical when the available data about agents in the swarm is limited. Our approach was proven to work even if using only the motion data of a single agent and replica, as the data from this agent implicitly contained enough information about the interactions in the swarm.

In this chapter, the model was explicitly represented by a set of parameters. The evolved parameters could thus be compared against the ground truth, enabling us to objectively gauge the quality of evolved models in two case studies as well as for 1000 randomly sampled behaviors. In principle, we could also evolve the structure of the agent's control system. The results of learning the agent's angle of view showed that our method may even learn the morphology of the swarming agents.

In collective behavior, abnormal agent(s) may have a great impact on the swarm [137]. For the same reason, the insertion of a replica that exhibits different behavior or is not recognized as con-specific may disrupt the global behavior and hence the models obtained may be biased. An appropriate strategy would be to isolate the influence of the replica. In particular, to evaluate a model one could perform two trials, one with only replicas each executing the model and the other with only agents. The data of the replicas and

agents from each trial could then be fed into the classifiers for making judgments. Some preliminary results suggest that there is no significant difference between either approach for the case studies considered in this chapter. In Chapter 2, we use *Turing Learning* to infer the behavior of a single agent. In all cases that are considered, either the agent or the replica is present.

2 Inferring Individual Behaviors Through Interactive Turing Learning

2.1 Introduction

In the previous chapters, we demonstrated how *Turing Learning* can be used for inferring swarm behaviors through observation. This is based on the implicit assumption that the behavioral repertoire of agents in the swarm can be fully revealed through observation. From the perspective of system identification, the target system then has high observability. However, when a target system has low observability [189], the agent's behavioral repertoire may not be fully revealed through observation. Moreover, only randomly generating the sequences of inputs to the system may not be insufficient too. The machine would then need to interact with the agent to explore its hidden information. Based on this idea, in this chapter we aim to infer agent behaviors that have low observability. In particular, we extend *Turing Learning* (as defined in Chapter 1) with interactive capability.

Observation and interaction are widely adapted by ethologists when investigating the behavior of animals [190, 191, 192]. When investigating animals in their natural habitats, passive observation is preferable as it is difficult to change the environmental stimuli. In this case, inferring the causal relations between the animal's behavior and its environmental stimuli may become challenging, since the stimuli are not under the observer's control. However, when the experiments are carried out in indoor laboratories, it is possible to change/control the stimuli to interact with the animals under investigation in a meaningful way. In [192], in order to investigate the cause of the dung beetle dance, biologists designed various experiments to interact with it and learn how it adapts to the environmental changes (e.g., appearance of disturbance or obstacles).

In this chapter, we investigate whether a machine could automatically infer agent behaviors (with low observability) through controlled interactions using the extended *Turing Learning* method. To validate this, two case studies are presented: deterministic agent behavior (Section 2.3.1) and stochastic agent behavior (Section 2.4.1). In these two case studies, the machine is able to control the agent's environmental conditions, which in this work corresponds to the intensity of the ambient light. At the same time, it is capable of simulating the actions of the agent. The learning result is a model of the agent that captures its behavior in relation to the environmental stimulus.

This chapter is organized as follows. Section 2.2 describes the methodology, illustrating how *Turing Learning* is extended to have interactive capability. The deterministic and stochastic behaviors under investigation are presented as two case studies (Sections 2.3 and 2.4). Section 2.3.1 describes the deterministic behavior. Section 2.3.2 presents the simulation setup. Section 2.3.3 presents the results of inferring the deterministic behavior, including analysis of the evolved models, the coevolutionary fitness dynamics, analysis of the evolved classifiers, a study showing the method's sensitivity to noise, and a comparison of *Turing Learning* with two metric-based methods—a single-population evolutionary approach and an approach based on coevolution of inputs and models. Section 2.4.1 describes the stochastic behavior (using a state machine) for the general case. Section 2.4.2 presents the simulation setup for inferring the stochastic behavior. Sections 2.4.3 and 2.4.4 present the obtained results for the case of 2 states and 3 states, respectively. Section 2.5 summaries the chapter.

2.2 Methodology

We extend *Turing Learning* (as described in Chapter 1) with interactive capability. In the following, we will describe the implementation that is related to the work in this chapter.

2.2.1 Models

The models are represented by a set of parameters that govern the rules of the agents. The details of these parameters will be described in Sections 2.3.1 and 2.4.1. As we

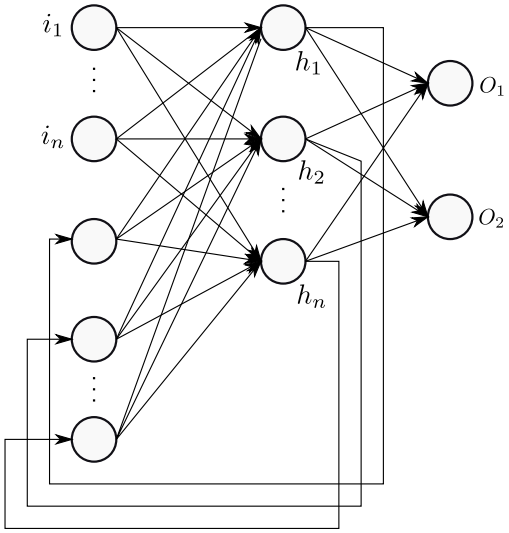


Figure 2.1: This diagram shows the structure of the classifiers used in this chapter. It is a recurrent Elman neural network [176] with i input neurons, h hidden neurons, and two output neurons (O_1 and O_2). O_1, which controls the stimulus, is fed back into the input; O_2 is used for making a judgment. A bias neuron with a constant input of 1.0 is connected to each neuron of the hidden and output layers. See text for details.

have argued in the previous chapters, explicit representation (i.e., evolving only the parameters) and knowing the ground truth (i.e., real parameters) makes it feasible for us to objectively gauge the quality of the models obtained.

2.2.2 Classifiers

Figure 2.1 shows the structure of the classifiers. The structure is similar to the one used in Chapter 1 (see Figure 1.1). However, the classifier (neural network) has an additional output that is used to control an environmental stimulus. Moreover, the environmental stimulus is also fed into the classifier as an additional input. In the following, we will explain how the classifier works.

Suppose the agent responds to the level of light intensity in the environment. We assume that the classifier can observe the agent's speed. The classifier (network) has two inputs. One is the light intensity in the environment at time step t, $I^{(t)} \in [0, 1]$, and the other is the speed $v^{(t)}$ of the agent. The speed of the individual for the classifier's

input is calculated by subtracting the previous estimated position from the current estimated position, and dividing the resulting number by the time interval between two measurements.

In order to make a judgment between a model and the agent, the classifier observes the behavior (speed) over a period of time. In addition, the classifier is also in control of the light intensity in the individual's environment. At time $t = 0$, the value of the light intensity is chosen randomly with a uniform distribution in the range $[0, 1]$. The neural network is then updated, using $I^{(0)}$ and $v^{(0)}$. The value of the light intensity for the next time step is obtained from the classifier's output neuron O_1, and the process repeats. After having iterated through all the time steps (a single trial), the final value of output neuron O_2 is used to make a judgment: the network decides on a model if $O_2 < 0.5$, and on the agent if $O_2 \geq 0.5$. The memory (value of hidden neurons) of the classifiers is reset at the end of every trial.

2.2.3 Optimization Algorithm

The algorithm used here is based on a $(\mu + \lambda)$ evolution strategy with self-adaptive mutation strengths [177, 109]. It is the same as the one used in Chapter 1. For the details of the implementation, see Section 1.1.1.3.

2.2.4 Fitness Calculation

Suppose the population sizes for the model and classifier are M and C, respectively. The fitness of each model is obtained by evaluating it with each of the classifiers in the competing population (C in total). For every classifier that wrongly judges the model as being the agent, the model's fitness increases by $\frac{1}{C}$. The final fitness is in $[0, 1]$.

The fitness of each classifier is obtained by using it to evaluate (i) each model in the competing population (M in total) once, and (ii) the agent L times with different initial light intensities. For each correct judgment of the model and the agent, the classifier's fitness increases by $\frac{1}{2 \cdot M}$ and $\frac{1}{2 \cdot L}$, respectively. The final fitness is in $[0, 1]$.

2.3 Case Study One

To validate our method, we present two case studies: one with deterministic behaviors and one with stochastic behaviors. The behaviors to be identified in this chapter were chosen to serve as a proof of concept study. While it may loosely correspond to how some animals react to the stimuli in their environment, it is not intended to mimic any specific animal. In these behaviors, non-trivial interaction with the agent is critical for leading the agent to reveal all of its behavioral repertoire.

2.3.1 Deterministic Behavior

We simulate a one-dimensional environment in continuous space. The simulation advances in discrete time steps $t \in \{0, 1, 2, \dots\}$. The (ambient) light intensity in the environment, I, can be varied continuously between 0 and 1. The agent distinguishes between three levels of light intensity, low ($0 \leq I < I_L$), medium ($I_L \leq I \leq I_H$), and high ($I_H < I \leq 1$). The two constants, I_L and I_H, represent the threshold of low and high levels of the light intensity, respectively. Hereafter, the three levels will be referred to as L, M, and H.

If the light intensity is at level M at time t, the speed of the agent, $s^{(t)} \in \mathbb{R}$, varies linearly with $I^{(t)}$ as:

$$s^{(t)} = k\left(I^{(t)} - 0.5\right), \tag{2.1}$$

where k is a constant.

We define two constants: $c_1 = k\,(I_H - 0.5)$ and $c_2 = k\,(I_L - 0.5)$. The deterministic behavior under investigation is shown in Figure 2.2. The agent's behaviors for levels L and H depend on the previous levels of light intensity (i.e. the agent has memory). The two behaviors are symmetrical to each other. Here, we will describe the behavior for level L; the behavior for level H is obtained by exchanging L with H and I_L with I_H in the following description.

When the light intensity is at level L, the agent's default speed is $k\,(I_L - 0.5)$ and remains at that value as long as the light intensity remains at level L. If the light intensity

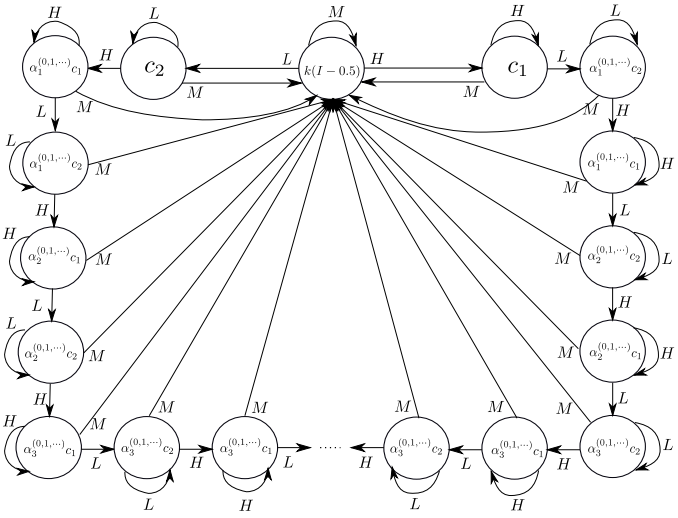


Figure 2.2: The deterministic behavior under investigation. It shows how the agent responses to the level of light intensity (L, M and H) in its environment. Each state represents the agent's speed. See text for details.

Table 2.1: This table shows the change of the agent's speed (shown in Figure 2.2), for an example sequence of light levels.

level	M	H	L	H	L	L	L	H	H	L	L
speed	$k(I-0.5)$	c_1	c_2	c_1	c_2	$\alpha_1^1 c_2$	$\alpha_1^2 c_2$	c_1	$\alpha_1^1 c_1$	c_2	$\alpha_2^1 c_2$

L	H	H	L	L	H	H	H	M	H	L	L
$\alpha_2^2 c_2$	c_1	$\alpha_2^1 c_1$	c_2	$\alpha_3^1 c_2$	c_1	$\alpha_3^1 c_1$	$\alpha_3^2 c_1$	$k(I-0.5)$	c_1	c_2	$\alpha_1^1 c_2$

is at level H (for any number of time steps), and then immediately changes to level L, and remains at that level for at least one more time step, then the agent's speed decays exponentially with a rate of α_1: that is, in the first time step that the light intensity is at level L, the agent's speed is $\alpha_1^0 k\,(I_L - 0.5)$; it then changes to $\alpha_1^1 k\,(I_L - 0.5)$, $\alpha_1^2 k\,(I_L - 0.5)$, and so on as long as the light intensity remains at level L. The agent now registers that α_1 has been activated. If another $H \to L \to L$ sequence is observed, the agent's speed now decays exponentially with a rate of α_2. If further $H \to L \to L$ sequences are observed, the exponential decay rate becomes and remains at α_3. Note that at any time, if the agent observes a light intensity at level M, its speed is proportional to the light intensity, as shown in Equation 2.1, and it forgets all its past observations

of the light intensity (i.e., the memory of the agent is reset).

The behavior of the agent can thus be represented by five cases; one where the agent's response to the light intensity is *proportional* (which occurs whenever the light intensity is at level M); one where the agent's response is *constant* (i.e. the agent's speed reaches the lower and upper saturation values (c_1 or c_2) as shown in Figure 2.2); and three where the agent's response *decays exponentially* with the decay rates α_1, α_2 and α_3, respectively.

Table 2.1 shows an example sequence of light levels, along with the corresponding speed of the agent (i.e., the speed shown in Figure 2.2).

Here, I_L and I_H are set to 0.1 and 0.9 respectively. k is set to 1.25; hence, the lower and the upper saturation values of the speed are $k\,(I_L - 0.5) = -0.5$ and $k\,(I_H - 0.5) = 0.5$. The exponential decay rates are set to: $\alpha_1 = 0.8$, $\alpha_2 = 0.4$, $\alpha_3 = 0.2$. Thus, in each case, the agent's speed decays exponentially towards zero. Note that these values (k, α_1, α_2 and α_3) are chosen arbitrarily and the coevolutionary algorithm is not sensitive to them.

The system identification task is to learn these four parameters (k, α_1, α_2 and α_3) of the agent. It is worth while to mention again that to learn α_3, the machine needs to at least output the sequence: $H \to L \to L \to H \to L \to L \to H \to L \to L$ or $L \to H \to H \to L \to H \to H \to L \to H \to H$. Since I_L and I_H are set to a relatively small and large value in $[0, 1]$ respectively, it is very unlikely to randomly generate such sequences.

2.3.2 Simulation Setup

We use three setups for the *Turing Learning* method. The setup, in which the classifier is in control of the light intensity in the agent's environment, is hereafter referred to as the "Interactive" setup. In order to validate the advantages of the interactive approach, we compared it against the situation where the classifier only observes the agent in a passive manner; that is, it does not control the light intensity in the environment. We considered two such setups: in the first setup (hereafter, "Passive 1") the light intensity is randomly chosen from the uniform distribution in $[0, 1]$, in every time step. In the

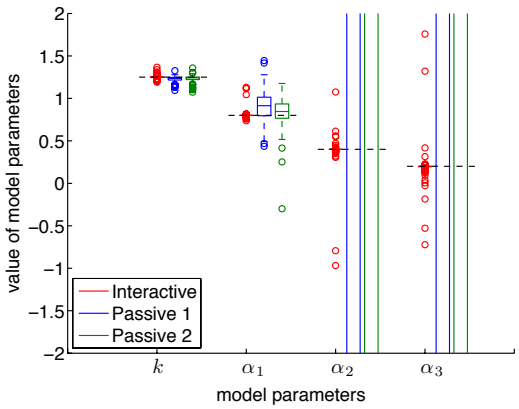


Figure 2.3: This plot shows the distributions of the evolved models with the highest subjective fitness in the 1000th generation in the coevolutions. Each box corresponds to 100 coevolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively. Note that in order to zoom in on the relevant range, some boxes and outliers are omitted from the plot.

second setup (hereafter, "Passive 2"), the light intensity is randomly chosen only at specific time intervals after a certain number of time steps elapsed (in this setup the number is chosen to be 10). All other aspects of these two setups are identical to the "Interactive" setup.

The population sizes of the models and classifiers are chosen to be 100, respectively. We performed 100 coevolution runs for each setup. Each coevolution run lasts 1000 generation. In one generation, each classifier conducts 100 trials on the agent. In each trial, the classifier observes the agent for 10 s at 0.1 s intervals, that is, a total of 100 data points.

2.3.3 Results

2.3.3.1 Analysis of Evolved Models

Figure 2.3 shows a box plot with the distributions of the evolved models with the highest subjective fitness in the 1000th generation over 100 coevolution runs of the three setups.

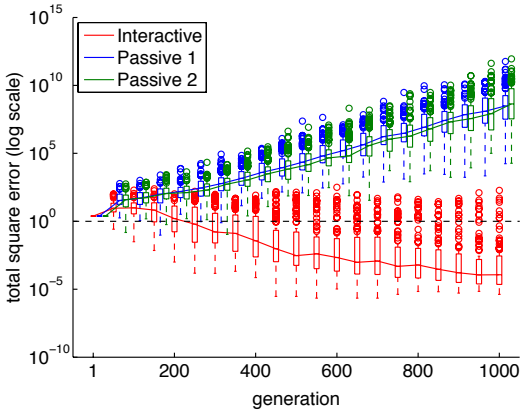


Figure 2.4: This plot shows the total square errors of the evolved model parameters compared to those of the agent over generations. The models with the highest subjective fitness in each generation are selected. Each box corresponds to 100 coevolution runs, and the solid lines correspond to the median error.

The passive coevolutions are able to evolve the parameters k and α_1 with a reasonable accuracy; however, they are not able to evolve α_2 and α_3. In the Passive 1 coevolution, the relative errors of the medians of the four evolved parameters $(k, \alpha_1, \alpha_2, \alpha_3)$ with respect to those of the agent are 1.2%, 14.3%, 7.8×10^4%, and 2.3×10^5%, respectively. The Passive 2 coevolution leads to similarly large relative errors in the evolved values of α_2 and α_3. This phenomenon can be explained as follows. If the light intensity changes randomly (either every time step, or every ten time steps), it is unlikely that the $H \to L \to L$ and/or $L \to H \to H$ sequences will occur enough times, without a level of M in between, such that the classifiers can observe the effects of α_2 and α_3. Therefore, the classifiers do not evolve the ability to distinguish the behavior of models from the behavior of the agent with respect to these two parameters, and in turn, these parameters do not converge to their true value in the model population.

In contrast to the passive coevolutions, the Interactive coevolution is able to evolve all the four parameters with a good accuracy. The relative median errors are 0.024%, 0%, 0.025% and 0.15% for k, α_1, α_2 and α_3 respectively. This implies that by the 1000^{th} generation, the classifiers have learned how to control the pattern of the light intensity in such a way that they can distinguish models from the agent based on the effect of any of the four parameters. Therefore, in order to compete for being selected in the

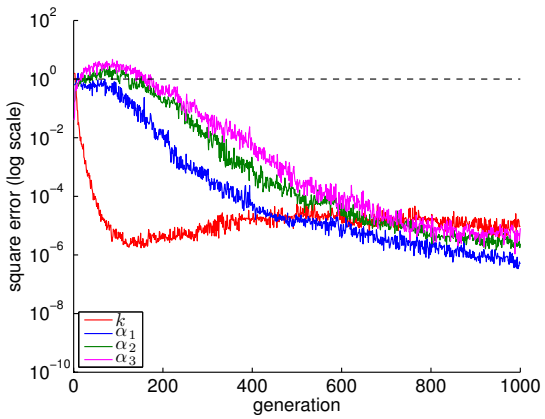


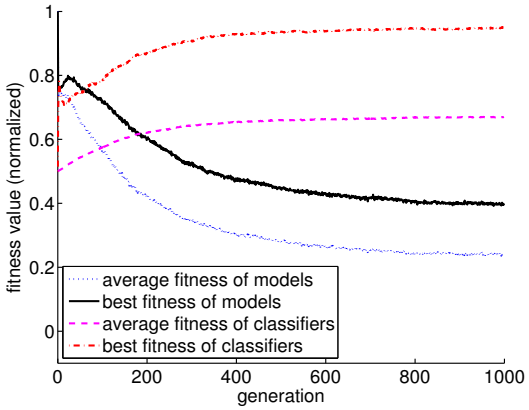
Figure 2.5: This plot shows how the square error in the individual model parameters changes over the generations in the Interactive coevolution. The curves correspond to median values from 100 coevolution runs.

population, the models are evolved to behave like the agent in every aspect.

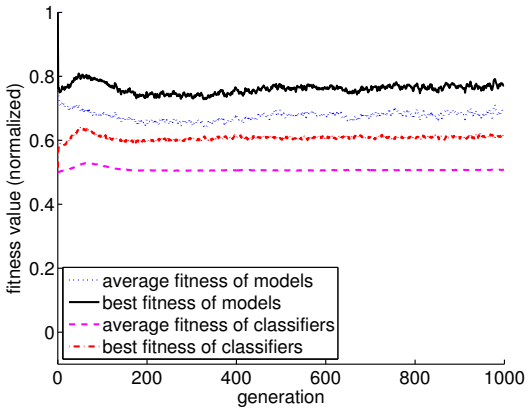
2.3.3.2 Coevolutionary Dynamics

Figure 2.4 shows the dynamics of the coevolutionary algorithms. The horizontal axis shows the generation, whereas the vertical axis shows the total square error of the model parameters, that is, the sum of the square errors in the four parameters (of the model with the highest subjective fitness in each generation) with respect to their true values. In the case of the Interactive coevolution, the median error starts to reduce after around the 100^{th} generation, and keeps decreasing until the last generation where it reaches a value of 10^{-4}. In contrast, in the case of the passive coevolutions, not only does the median error not decrease, but it increases to a value of 10^{8} by the 1000^{th} generation.

We now analyze how the four individual parameters evolve during the course of the Interactive coevolution, which is the only fully-successful setup. The plot shown in Figure 2.5 reveals how the learning proceeds in the coevolution. Parameter k is the first to be learnt, followed by α_1, while parameters α_2 and α_3 take a longer time to approximate the true values. This means that the classifiers first learn to distinguish models from the agent on the basis of k and α_1. This ability of the classifiers drives the model population to evolve k and α_1, in order to mislead the classifiers. Eventually, the classifiers also



Interactive coevolution



Passive 1 coevolution

Figure 2.6: This plot shows the subjective fitness (normalized) of the classifiers and the models in (a) the Interactive coevolution, and (b) the Passive 1 coevolution. The curves show the average fitness across 100 coevolution runs.

learn to exploit the effects of α_2 and α_3 in order to make the right judgment; thereby driving the model population to evolve these two parameters accurately. After about the 600^{th} generation, the learning of the four parameters proceeds with approximately identical rates.

In order to analyze why the Interactive coevolution is successful while the passive ones are not, we can look at the dynamics of the subjective fitnesses of the classifiers and the models (as defined in Section 2.2.4) during the course of the coevolution. As both of the passive coevolutions fail to converge, we present the analysis of fitness dynamics only for Passive 1 coevolution (the other case was found to have similar dynamics). Figure 2.6 shows the fitness dynamics of the Interactive and the Passive 1 coevolutions. In the case

of the Interactive coevolution (see Figure 2.6(a)), the average fitness of the classifiers starts off at 0.5, which means that the classifiers make judgments that are no better than random judgments. However, the classifiers quickly improve in fitness, which in turn causes the fitness of the models to decrease. This increases the selective pressure on the models. After about 600 generations both the fitness of classifiers and models reach a steady state, which according to Figure 2.5 corresponds to the region where the four parameters evolve with virtually identical rates. In the case of the Passive 1 coevolution (see Figure 2.6(b)), the average fitness of the classifiers also starts off at 0.5. In the first few generations, this increases slightly, because the classifiers learn how to distinguish models from the agent on the basis of parameters k and α_1. However, the models quickly adapt to this new ability of the classifiers. Now, as the classifiers are unlikely to have the opportunity to observe the effects of α_2 and α_3, their average fitness returns to 0.5. This leads to a disengagement phenomenon, in which there is no more meaningful selection in the model population, therefore leading the parameters α_2 and α_3 to drift, the effect of which can be seen in Figure 2.4.

2.3.3.3 Analysis of Evolved Classifiers

In this section, we analyze the evolved classifiers in the Interactive coevolution. The model described in Section 2.2.1 is defined by four parameters: $\{k, \alpha_1, \alpha_2, \alpha_3\}$. In order to evaluate the quality of the evolved classifiers we performed a grid search over the space of the amount of disturbance (i.e. noise) injected into each of the four parameters. We used 11 noise magnitude settings per parameter, $M \in \{0, 0.1, \ldots, 1\}$, with the noise being added to the parameters as follows:

$$p' = p(1 + \mathcal{U}(-M, M)), \tag{2.2}$$

where $p \in \{k, \alpha_1, \alpha_2, \alpha_3\}$ represents any of the four parameters, and $\mathcal{U}(-M, M)$ denotes a uniform distribution on the interval $(-M, M)$. Note that $M = 0$ corresponds to no noise, whereas $M = 1$ means that the noisy value of the parameter can be between 0 and twice its actual value.

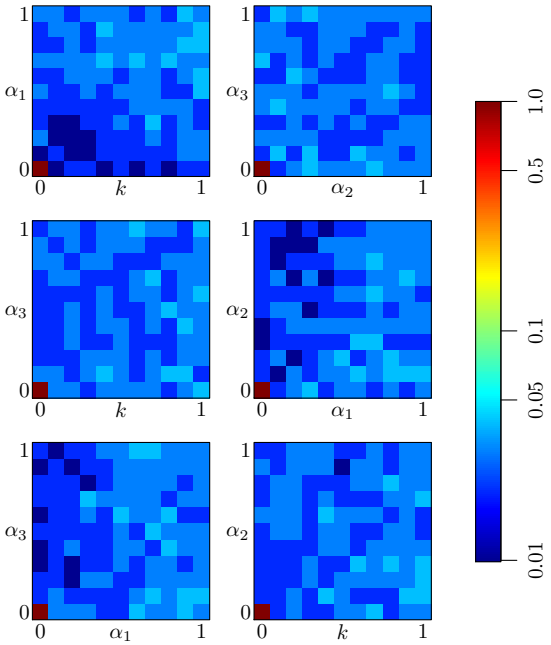


Figure 2.7: This figure shows the landscape of U^* (log scale) for the overall best classifier over the six sub-spaces with two parameters as degrees of freedom (from 100 coevolution runs). Each axis in each plot ranges between 0 and 1, corresponding to the minimum and maximum magnitude of noise added into each parameter respectively. See text for details.

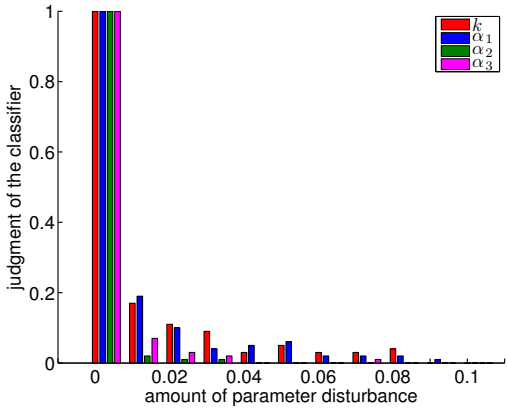


Figure 2.8: This plot shows the average judgment [U, see Equation 2.3] of the best evolved classifier over 100 trials when the magnitude of noise added into each parameter of the agent is within 0.1.

For the sake of simplicity, we only analyzed the classifiers with the highest subjective fitness in the last generation of the 100 coevolutions. For each of the 100 classifiers, and

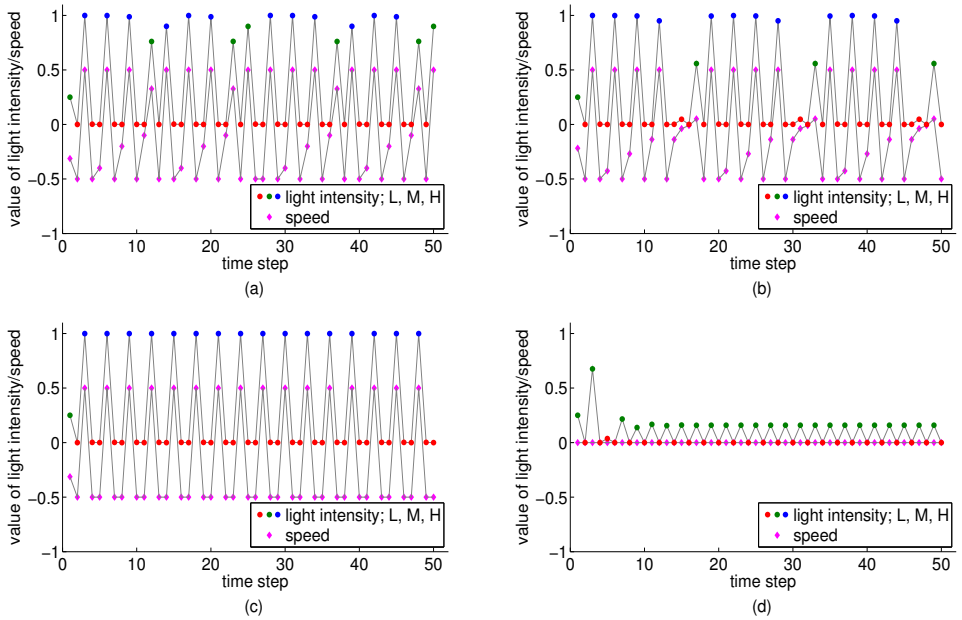


Figure 2.9: This plot shows the light intensity sequences as output by the overall best classifier (circular points), along with the corresponding speeds of the agents (diamond-shaped points) in four trials conducted on different agents: (a) the agent; (b) a model similar to the agent, generated randomly according to Equation 2.2 with $M = 0.5$; (c) a model whose speed is linear to the light intensity; and (d) a static model, whose speed is always 0 (i.e. $k = 0$). The colors of the circular points (red, green, blue) correspond to light intensities at levels L, M and H, respectively.

for each combination of noise magnitudes, we conducted 100 trials. In other words, we conducted $100 \cdot 11^4 \cdot 100 = 146,410,000$ trials in total. In each trial, we modulated the agent's parameters according to Equation 2.2. Each trial was run for T = 100 time steps (the same setting used within the coevolutions). For each combination of magnitudes employed, the overall performance U was computed as the sum of the final judgments of the classifier in each trial, divided by the total number of trials, N:

$$U = \sum_{i=1}^{N} J_i / N, \tag{2.3}$$

where, $J_i \in \{0, 1\}$ is the final judgment of the classifier in trial i, and N is the total number of trials conducted. Note that $J_i = 0$ and $J_i = 1$ imply that the classifier has judged the behavior as being that of a model and the agent, respectively.

In order to find the overall best classifier from the 100 classifiers analyzed, we defined the following metric:

$$W = [1 - U(0,0,0,0)] + \frac{1}{\Omega} \sum_{\substack{i,j,k,\ell \in \{0,0.1,\ldots,1\} \\ i^2+j^2+k^2+\ell^2 \neq 0}} \sum \sum \sum \sum (i+j+k+\ell)\, U(i,j,k,\ell)\,, \tag{2.4}$$

where $\Omega = 29282$ is the maximum value that the quadruple sum can achieve (i.e. if all the U's are equal to 1). The first term in Equation 2.4 penalizes the classifier if $U < 1$ when there is no noise on the parameters; they are thus undisturbed and identical to the ones of the agent. The second term penalizes the classifier if $U > 0$ for any non-zero noise combination, with increasing penalties being applied to higher noise magnitudes. The normalization of the second term by Ω serves to make the two terms contribute equally to W, which can take values in $[0, 2]$. Note that the minimum value of $W = 0$ can only be achieved by the perfect classifier, that is, one that outputs 1 if $i = j = k = \ell = 0$ and 0 otherwise. In our case, the best classifier achieved a value of $W = 3.7 \cdot 10^{-3}$.

The performance landscape of the models is 5-dimensional (4 model parameters plus performance measure), and cannot be visualized directly. Therefore, we considered each combination of two model parameters ($\binom{4}{2} = 6$ combinations) as a sub-space, and for each point in this sub-space, we calculated the performance measure as the maximum value over the sub-space spanned by the remaining two parameters. For instance, on the sub-space (k, α_1), the performance measure $U^*(k, \alpha_1)$ was calculated as:

$$U^*(k, \alpha_1) = \max_{\alpha_2, \alpha_3} U(k, \alpha_1, \alpha_2, \alpha_3)\,. \tag{2.5}$$

Note that for all the points except $(0, 0, 0, 0)$, Equation 2.5 corresponds to the worst-case scenario for the classifiers, because the value of U for the ideal classifier at these points is 0. For the point $(0, 0, 0, 0)$, the value of U for the ideal classifier is 1.

Figure 2.7 shows the landscape of U^* (log scale) for the overall best classifier over the six sub-spaces. When interacting with the agent (i.e. $i = j = k = \ell = 0$), the output of the classifier is always 1. This corresponds to the point $U^*(0, 0)$ in the six sub-spaces of Figure 2.7 (here only the maximum is shown). When interacting with the models

(i.e. when the parameters of the agent are perturbed), for any combination of noise magnitudes, the average output of the classifier is below 0.05 for the six sub-spaces.

In order to further analyze how sensitive the overall best classifier is when the parameters of the agent are only slightly perturbed, we set the maximum magnitude of noise added to each parameter to 0.1, and used a resolution of 0.01. In this evaluation, we only injected noise into one parameter at a time. For each noise magnitude, the classifier was evaluated in 100 trials (with different initial light intensities, and randomly-generated noise), and the average performance measure was computed according to Equation 2.3.

Figure 2.8 shows the average judgment [U, see Equation 2.3] of the best classifier, which was evaluated in 100 trials. With increasing noise magnitudes, the average judgment of the classifier approaches zero, which means that the classifier can identify the agents as models more consistently. The classifier has different sensitivities to the four parameters; the effects of noise on α_2 and α_3 on its judgment are higher than those of k and α_1. For instance, in the case of α_2, even with $M = 0.01$, the classifier correctly judges the behavior as being that of a model in 98% of the trials. As we have seen, k and α_1 are the first two parameters to be identified in the coevolution. α_2 and α_3 are addressed in the later generations of the coevolutionary process, and it seems that the classifier is more sensitive to these two parameters.

We now investigate how the overall best classifier interacts with the agents. We conducted four trials with four different agents: (a) the agent; (b) a model similar to the agent, generated randomly according to Equation 2.2 with $M = 0.5$; (c) a model whose speed is linear to the light intensity without exponential decay (i.e. $k = 1.25$, $\alpha_1 = \alpha_2 = \alpha_3 = 1$); and (d) a static model, whose speed is always 0 (i.e. $k = 0$). In each trial, the initial light intensity was set to 0.25. The trials lasted for 100 time steps.

Figure 2.9 shows the sequences of light intensity output by the classifier, along with the speed of the agents in the four trials, for the first 50 time steps (we observed that the last 50 time steps constitute a repetition of the first 50). As we can see, the classifier outputs different sequences of light intensity in order to interact with the different agents. The greater the difference between a model and the agent, the more varied is the sequence of light intensities that the classifier outputs when interacting with it as compared to the one it outputs when interacting with the agent. For the agent, the classifier repeatedly outputs a $H \rightarrow L \rightarrow L$ sequence, in order to fully reveal the behavior (this is analyzed

in detail in the following paragraph). For the model that is similar to the agent (see Figure 2.9(b)), the sequence of light intensities is similar to the one produced for the agent, but it is not identical. Interestingly, for the model without exponential decay (see Figure 2.9(c)), the classifier produces the $H \to L \to L$ sequence even more often than it does for the agent, but it does not set the light intensity to level M. For the static model (see Figure 2.9(d)), the classifier never outputs a $H \to L \to L$ sequence; instead, it outputs a sequence that alternates between M and L.

We now focus on analyzing the behavior of the agent in Figure 2.9(a). Initially, the classifier outputs the sequence $L \to H \to L \to L \to H \to L \to L \to H \to L \to L \to M$, which means that by the 12^{th} time step, it has already observed the effects of all four parameters. Interestingly, the classifier then (essentially) repeats this sequence and thus observes the effect of all the parameters for another seven times (four repetitions occur in the last 50 time steps, which are not shown in Figure 2.9(a)). We conjecture that the repetitions make the classifier more robust in determining whether the behavior under observation is that of a model or the agent. These results suggest that the classifier succeeds in controlling the level of ambient light in a way that helps distinguish between the agent and the models, revealing all aspects of the underlying behavior. Note that the classifiers are not provided with any prior information about the agent, including its structure. Thus, all knowledge is gained through an evolutionary process that is driven by the accuracy of their judgments.

2.3.3.4 Noise Study

We now consider the situation where, during the coevolutionary process, noise is injected into the agent's behavior, and the agent's positions as measured by the system. This makes the overall setup more realistic as the locomotion of agents and any tracking system will be affected by noise and measurement error. Since the passive coevolutions fail even in the noiseless case, we consider only the Interactive coevolution for the sake of simplicity. We performed 100 coevolutionary runs with the following settings. The light intensity perceived by the agent at time t is obtained by multiplying the actual intensity by a random number generated uniformly in $(0.95, 1.05)$, and capping the perceived intensity to 1 if it exceeds this value. Noise is also applied to the speed of the

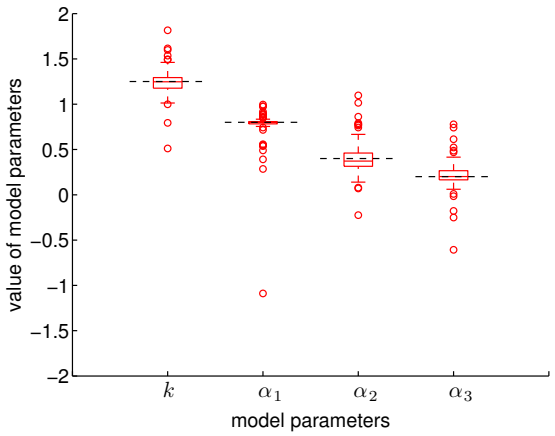


Figure 2.10: This plot shows the distributions of the evolved models with the highest subjective fitness in the 1000$^{\text{th}}$ generation of the Interactive coevolution with noise (for a comparison to the case without noise, see Figure 2.3). The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively.

agent by multiplying the original speed with a random number generated uniformly in $(0.95, 1.05)$. Noise on the estimated position on the agent is applied by adding a random number generated from a normal distribution: $\mathcal{N}(0, 0.005)$.

Figure 2.10 shows a box plot with the distributions of the evolved models with the highest subjective fitness in the 1000$^{\text{th}}$ generation of the Interactive coevolution with noise. The effect of the noise is to widen the distribution of the evolved parameters across the 100 coevolutionary runs; however, the median values of the evolved parameters are still very close to the true values. Interestingly, the Interactive coevolution does not seem to learn α_2 and α_3 significantly worse than it does learn k and α_1.

2.3.3.5 Using a Single-Population Evolutionary Algorithm

In order to compare *Turing Learning* against a more traditional approach, we used a simple evolution where a single population of models evolves. We call this method SP-EA. As there are now no classifiers, an interactive approach is not possible, and thus we conducted 100 evolutionary runs for the Passive 1 and Passive 2 methods of changing the

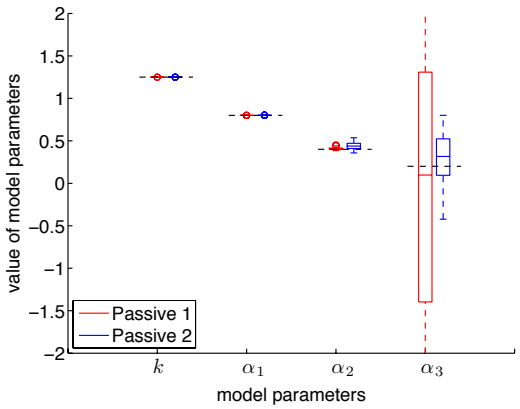


Figure 2.11: This plots shows the distributions of the evolved models with the highest fitness in the 100000^{th} generation in the simple evolutions with a single population. Each box corresponds to 100 evolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively. Note that in order to zoom in on the relevant range, some boxes and outliers are omitted from the plot.

Table 2.2: This table shows a comparison of all approaches. The numbers show the relative errors of the evolved parameters (median values over 100 runs) with respect to the parameters of the agent (in absolute percentage). TL: *Turing Learning*.

	k	α_1	α_2	α_3
Interactive (TL)	0.024	0	0.025	0.15
Passive 1 (TL)	1.2	14.3	7.8×10^4	2.3×10^5
Passive 2 (TL)	0.7	5.74	1.3×10^4	3.3×10^5
Passive 1 (SP-EA)	0	0	1.2	48.7
Passive 2 (SP-EA)	0	0	9.8	48.5
CEA-IM	4.0×10^{-5}	4.9×10^{-5}	1.1×10^{-3}	2.5×10^{-4}

light intensity in the agent's environment. The structure of the evolution is identical to the sub-algorithms used in the coevolution, except for the fitness evaluation step. Now, in each generation, 100 experiments are performed on the agent using 100 randomly generated intensity patterns. The 100 intensity patterns are used to evaluate a model 100 times. The average square error between the model's and the agent's speed sequences is used as the model's fitness. Each evolutionary run lasts $100,000$ generations. In other words, the number and duration of experiments on the agent is kept the same as that

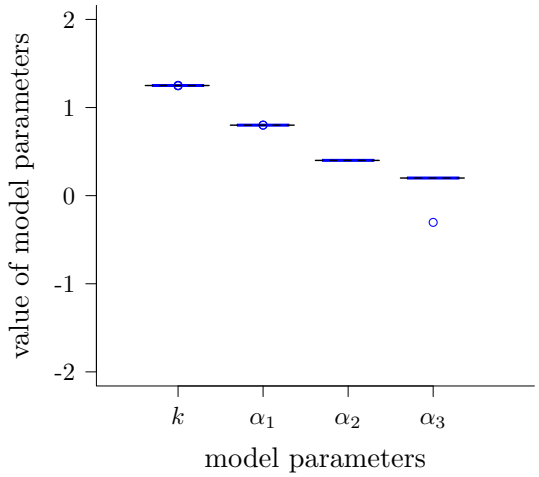


Figure 2.12: This plots shows the distributions of the evolved models with the highest fitness in the 100000^{th} generation in the simple evolutions with a single population. Each box corresponds to 100 coevolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively.

in the coevolutionary approach, as outlined in Section 2.3.2.

Figure 2.11 reveals that the evolution is able to identify parameters k, α_1, α_2, but not α_3. Note that, apart from the single-population evolution not being able to consistently identify the parameter α_3, they also rely on a pre-defined metric for their operation; in this case, computed as the square error between the model's and the agent's speed sequences.

2.3.3.6 Coevolution of Inputs and Models

In this section, we compare *Turing Learning* with another coevolutionary system identification method, in which inputs and models competitively coevolve. In particular, we use the classifiers to generate sequences of inputs (in this case, light intensity). The models are optimized through minimizing the square error of speed between the agent and models, given the same sequence of inputs generated by the classifiers. The clas-

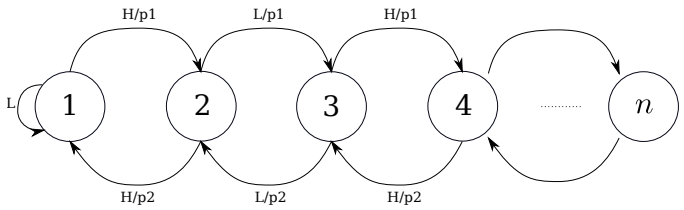


Figure 2.13: A state machine which represents the stochastic behavior of the agent under investigation. The initial state is 1. p_1, p_2 are probabilities. The ambient light intensity that the agent responds to has two levels: H and L. See text for details.

sifiers compete with the models through generating inputs that cause the maximum disagreement between the model's prediction and the agent's output, which is similar to the concept in [14]. Note that in this case the classifiers are only used for generating the inputs and they do not make judgments. We call this method CEA-IM. Figure 2.12 shows the results of CEA-IM. As we can see, the model parameters are also identified with a high accuracy. In other words, there is no 'true' interaction between the agent and classifiers during the experiments in *Turing Learning* when investigating the deterministic behavior. The classifiers only need to learn how to generate a fixed sequence of inputs to extract all the information from the agent. For a comparison of all approaches, see Table 2.2.

In order to further validate and highlight the benefit of *Turing Learning*, we investigate the stochastic behaviors in the following section.

2.4 Case Study Two

2.4.1 Stochastic Behavior

Stochastic behaviors are widely observed in the animal kingdom. Given the same stimuli, the animal may behave differently. In order to make the animal under investigation reveal all its behavioral repertoire, ethologists sometimes need to interact with the animals in real time and change the stimuli dynamically according to the animal's response. Based on this motivation, in this section we apply *Turing Learning* to infer stochastic behaviors

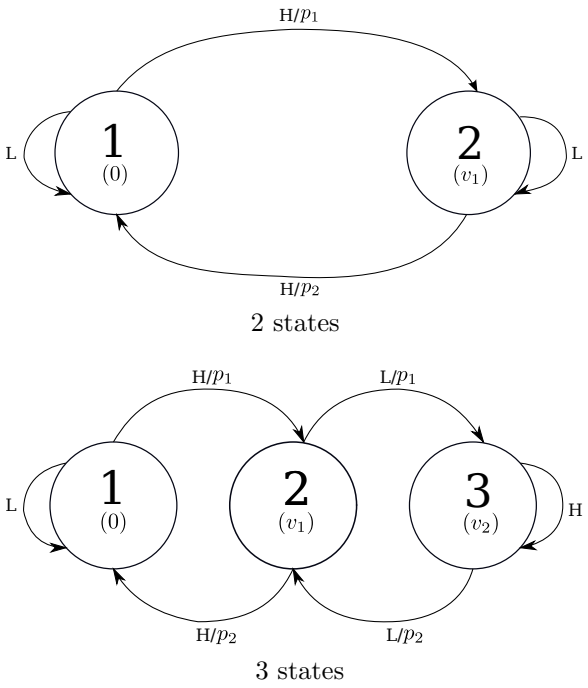


Figure 2.14: The stochastic behaviors with (a) 2 states and (b) 3 states under invesitgation. p_1 and p_2 are probabilities. See text for details

and investigate whether a machine could interact with the agent through dynamically changing the stimulus rather than applying a fixed sequence.

We will describe the stochastic behavior for the general case. It is represented by a state machine, shown in Figure 2.13. Suppose the agent has n states. The agent's behavior depends on the level of the light intensity in the environment and its current state (i.e., the agent has memory). For the initial state 1, if the light intensity is low (L), the agent stays in state 1; if the light intensity is high (H), the agent moves forward to state 2 with probability p_1. For states $i = 2, 4, 6, \cdots, 2\lfloor\frac{n-1}{2}\rfloor$, if light intensity is L, the agent moves to state $(i+1)$ with probability $p1$; if the light intensity is H, the agent moves to state $(i-1)$ with probability $p2$. The behavior for states $i = 3, 5, 7, \cdots, 2\lfloor\frac{n}{2}\rfloor - 1$ is obtained by exchanging L with H in the above description. When the agent is in state n, its behavior is similar to that in states $i = 2, 3, 4, n-1$. The only difference is that the agent can not move to a higher state. Figures 2.14(a) and 2.14(b) show the agent behavior with 2 and 3 states, respectively. These are the two behaviors to be investigated in the thesis.

For the agent's behavior, p_1 is set to a low value and p_2 is set to a high value. As a consequence of this choice, the agent has a low chance of moving forward to a state with a higher number and thus the higher state has lower observability. The classifiers need to learn how to interact with the agent to infer all its behavioral repertoire.

For a proof of concept study, we assume that the agent moves in one-dimensional space and moves at a constant but different speed in each state. Suppose the agent is initially static (zero speed) and its initial state is known. The system identification task is then to identify the parameters of the other states (i.e., speed values v_1 and v_2 shown in Figure 2.14) and the two probabilities, p_1 and p_2. The speed of the agent in each state is chosen arbitrarily. p_1 and p_2 are chosen to have a value of 0.1 and 1.0. As explained, this makes the higher states harder to be observed.

In order to make the agent stay in a higher state, the classifiers need to capture the moment when the agent reaches that state, and toggles/switches the level of light intensity immediately. This makes it easier for *Turing Learning* to learn the behavior as the state can be observed for a sufficient time. If the classifiers fail to do that, the agent would move immediately back to the previous state.

For the 2-state machine shown in Figure 2.14(a), v_1 is selected to be 0.5; for the 3-state machine shown in Figure 2.14(b), v_1 and v_2 are selected to be 0.5 and 1.0, respectively. Therefore, the parameters to be identified for the 2-state and 3-state agent behavior are:

$$\mathbf{q}_1 = (v, p_1, p_2) = (0.5, 0.1, 1.0). \tag{2.6}$$

$$\mathbf{q}_2 = (v_1, v_2, p_1, p_2) = (0.5, 1.0, 0.1, 1.0). \tag{2.7}$$

2.4.2 Simulation Setup

To evaluate *Turing Learning*, we still used three setups: "Interactive", "Passive 1" and "Passive 2", as discussed in Section 2.3.2. We also compared *Turing Learning* with the two metric-based methods (SP-EA and CEA-IM) described in Sections 2.3.3.5 and 2.3.3.6, respectively. For each setup, we added a certain amount of noise into the

measurement of speed. This is realized by multiplying the agent's real speed with a random value in $[0.95, 1.05]$ in each time step.

2.4.3 Results: Two States

In this section, we present the results of inferring the 2-state agent behavior shown in Figure 2.14(a), using the setups in Section 2.4.2.

2.4.3.1 Analysis of Evolved Models

Figure 2.15 shows a box plot with the parameters of the evolved models with the highest subjective fitness in the 1000^{th} generation for (a) *Turing Learning*, (b) SP-EA, and (c) CEA-IM. Using *Turing Learning*, the system identified all parameters of the agent with good accuracy. For the other two metric-based methods, all the three parameters are not learned well. Instead, the three evolved parameters converge into three different values: $v_1 \rightarrow 0.0$, $p_1 \rightarrow 1.0$, $p_2 \rightarrow 0.0$. The failure of SP-EA and CEA-IM can be explained as follows. As the behavior is stochastic, given the same sequence of inputs the agent would probably exhibit different behaviors. Therefore, quantitatively measuring the difference between the models and agent (e.g., using square error) would not lead the model parameters to converge into their true values. For all methods, the means (standard deviations) of the AEs of each parameter of the evolved model with the highest fitness in the final generation over 30 coevolution runs are shown in Table 2.3. Clearly, *Turing Learning* infers the stochastic behavior significantly better than the two metric-based methods. There is no significant difference among "Interactive", "passive 1" and "passive 2" setups of *Turing Learning* in terms of AEs of the evolved parameters.

In order to show the advantage of "Interactive" setup of *Turing Learning*, we investigate the convergence of model parameters during the evolutionary process. Figure 2.16 shows the convergence of the model parameters over generations for the three setups of *Turing Learning*. As we can see, the evolved model parameters in the "Interactive" setup converge much faster than those in the two passive setups. For the "Interactive" setup, after about 100 generations, all the three parameters converge into their true values. For

Table 2.3: This table shows a comparison of all approaches for learning the 2-state stochastic behavior shown in Figure 2.14(a). The values show means of AEs (defined in Equation 1.9) of each evolved model parameter with respect to that of the agent. TL: *Turing Learning*.

	v_1	p_1	p_2
Interactive (TL)	0.003	0.03	1.6×10^{-5}
Passive 1 (TL)	0.01	0.03	1.0×10^{-5}
Passive 2 (TL)	0.02	0.02	1.0×10^{-5}
Passive 1 (SP-EA)	0.47	0.9	1.0
Passive 2 (SP-EA)	0.47	0.9	1.0
CEA-IM	0.47	0.9	1.0

the "Passive 1", all the parameters converge after about 200 generations, while for the "Passive 2" it takes longer time. In terms of v_1, there is a much smaller disturbance in the "Interactive" setup than that in the other two setups.

2.4.3.2 Analysis of Evolved Classifiers

In order to investigate why the "Interactive" setup of *Turing Learning* learns the agent behavior faster than the two passive setups, we post-evaluated how the classifiers interact with the agent during a trial.

Figure 2.17 shows how the classifier with the highest subjective fitness in different generations (of a particular coevolution run) interact with the agent in a trial. A similar phenomenon could be observed in other coevolution runs. As shown in the top left of Figure 2.17, the classifier outputs H level. It waits until the agent 'jumps' from state 1 to state 2 (a stochastic process), and then immediately switches the light intensity from H to L level in order to make the agent stay in state 2 as long as possible. Therefore, the agent's behavior in state 2 (hidden information) can be observed longer and inferred efficiently through the process of evolution. Note that the classifier has learned a good strategy and exhibited such 'intelligent' behavior at the very beginning of the coevolution run (before 50 generations in this case). After 500 generations, the classifier slightly changed the strategy. Instead of always outputting H level, it keeps switching between H and L level until it observes the agent 'jumps' from state 1 to state 2, and after that

Table 2.4: This table shows a comparison of all approaches for inferring the 3-state agent behavior shown in Figure 2.14(b). The values show means of AEs (defined in Equation 1.9) of each evolved model parameter with respect to that of the agent. TL: *Turing Learning*.

	v_1	v_2	p_1	p_2
Interactive (TL)	0.005	0.03	0.02	2.0×10^{-6}
Passive 1 (TL)	0.02	0.23	0.05	0.03
Passive 2 (TL)	0.02	0.47	0.08	0.13
Passive 1 (SP-EA)	0.47	0.97	0.9	1.0
Passive 2 (SP-EA)	0.47	0.98	0.9	1.0
CEA-IM	0.46	0.96	0.67	0.89

it immediately switches the light intensity from H to L level. This is unlikely to happen when generating random sequences of inputs.

2.4.4 Results: Three States

In order to further demonstrate the advantage of "Interactive" setup of *Turing Learning*, this section discusses the results of inferring the 3-state stochastic agent behavior shown in Figure 2.14(b).

2.4.4.1 Analysis of Evolved Models

Figure 2.18 shows the distribution of the evolved models in the 1000^{th} generation for all setups. The "Interactive" setup of *Turing Learning* is the only one that infers all the parameters of the agent with good accuracy. For the two setups using pre-defined metrics (SP-EA and CEA-IM), all the parameters are not well learned. Table 2.4 compares the accuracy of each parameter of the evolved model with the highest fitness in the final generation over 30 coevolution runs using all approaches.

Figure 2.19 shows the evolutionary process of the models for the three setups of *Turing Learning*. As we can see, all the model parameters in the "Interactive" setup converged to their true value smoothly within about 200 generations. For the two passive setups,

v_1, p_1 and p_2 are well learned, but they still take longer time to converge to their true values than those of the "Interactive" setup. There is a dramatic disturbance during the evolutionary process of v_2 for the passive setups, and this parameter is not well learned.

2.4.4.2 Analysis of Evolved Classifiers

Figure 2.20 shows an example of how the classifier with the highest subjective fitness in different generations (of a particular coevolution run) evolved to interact with the agent. In other coevolution runs, we observed a similar phenomenon. As shown in Figure 2.20, the strategy learned by the classifiers shown in 3 of the 4 sub-figures (corresponding to the 100^{th}, 200^{th} and 1000^{th} generations) is as follows. The classifier outputs H first, and once the agent moves forward from state 1 to state 2, the classifier switches the light intensity into level L and keeps it in that level. As long as the agent moves forward from state 2 to state 3, the classifier switches the light intensity from L to H and keeps the light intensity in that level. Note that the 'best' classifier sometimes lost its ability to interact with the agent in the 500^{th} generation as shown in bottom left of Figure 2.20. However, this does not influence the learning process, as long as there are still some other classifiers in the population obtain this interactive ability.

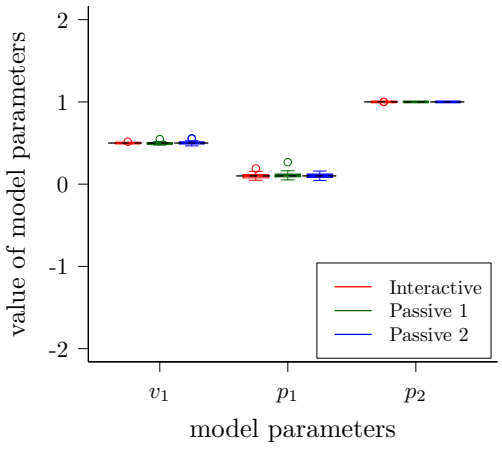
2.5 Summary

In this chapter, we extend *Turing Learning* with interactive ability to autonomously learn the behavior of an agent. We have shown that, by allowing classifiers to control the stimulus in the agent's environment, the system is able to correctly identify the parameters of relatively complex behaviors. The advantage of *Turing Learning* with interaction is validated using two case studies: stochastic and deterministic behaviors of an agent. In both case studies, the results show that learning through interaction can infer the agent behaviors better or faster than only through passive observation.

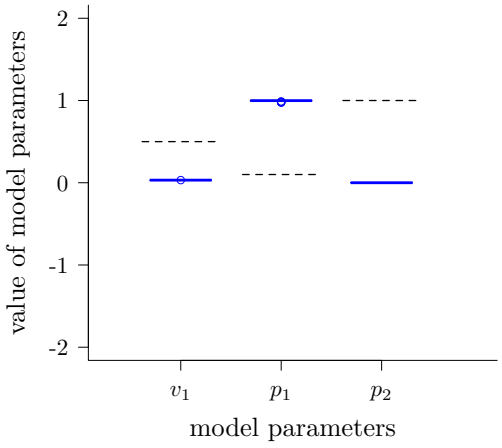
When inferring the deterministic behavior, the classifiers have learned to generate a complex but static sequence of inputs to extract all the hidden information from the

agent, which facilitates the learning process. However, generating such sequences at random is unlikely. This makes the SP-EA (in which the inputs are generated randomly) fail to infer all the parameters of the agent. However, by coevolving inputs and models (CEA-IM), the system still obtained very good model accuracy in terms of all parameters.

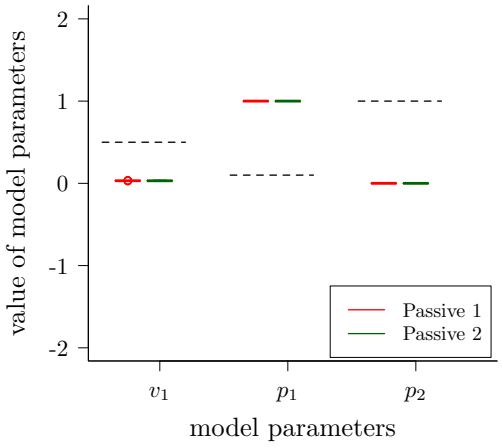
When inferring the stochastic behavior, the advantage of *Turing Learning* with interaction becomes obvious. It performs significantly better than the two metric-based methods (SP-EA and CEA-IM) in inferring the behaviors. The classifiers learned to dynamically interact with the agent, which made it possible to infer all the agent's parameters in an efficient way. In a sense, the machine, using *Turing Learing*, exhibits an 'intelligent' behavior known from humans: it adapts to interact with a system in order to learn faster what the system does. This property of *Turing Learning* could pave the way to further development of machine intelligence.



Turing Learning

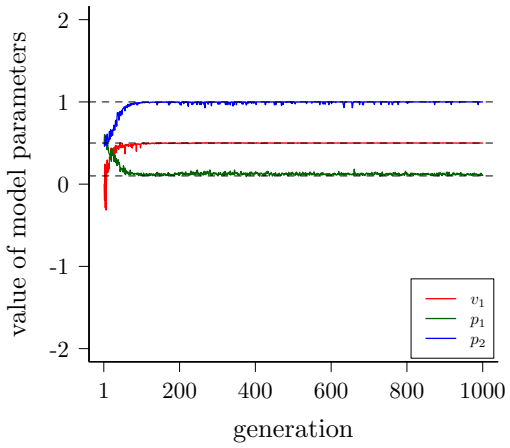


SP-EA

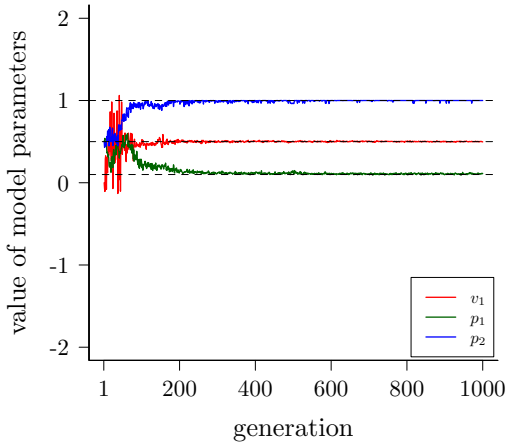


CEA-IM

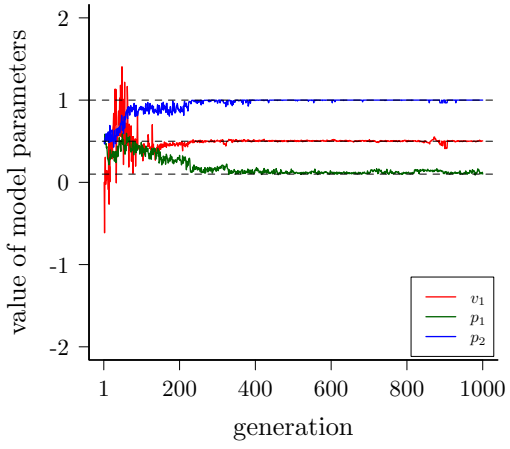
Figure 2.15: This plot shows the distributions of the evolved models with the highest subjective fitness in the 1000^{th} generation in the coevolutions. Each box corresponds to 30 coevolution runs. The dotted lines correspond to the values of the three parameters that the system is expected to learn (i.e. those of the agent). See text for details.



Interactive



Passive 1



Passive 2

Figure 2.16: Evolutionary process of the evolved model parameters for (a) “Interactive”, (b) “Passive 1” and (c) “Passive 2” setups of the metric-free method when learning the 2-state agent behavior. Curves represent mean values across 30 coevolution runs. Dotted black lines indicate true values.

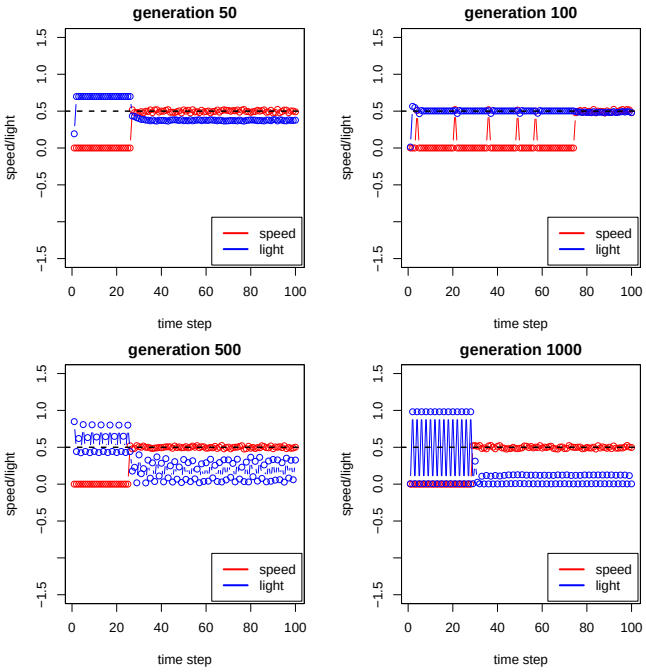
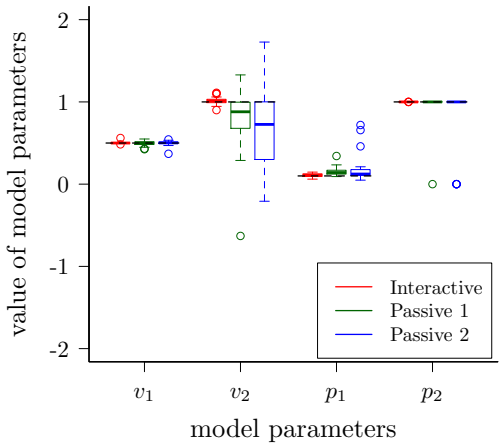
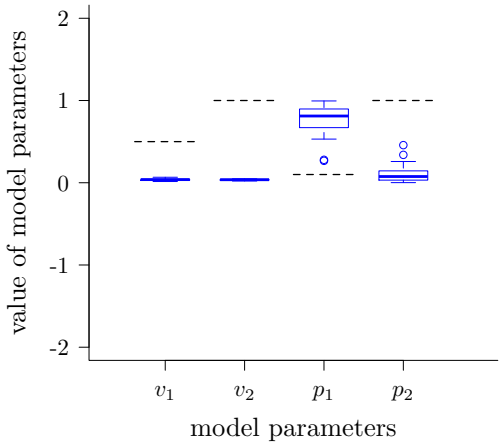


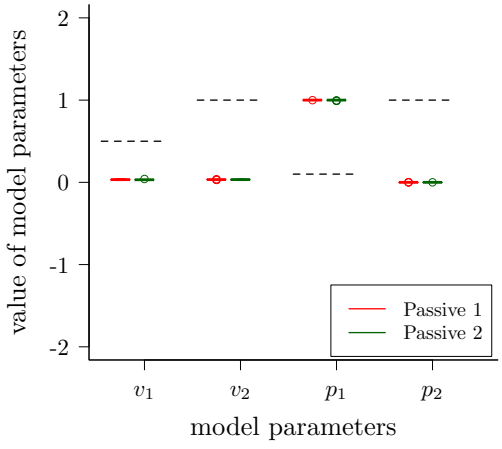
Figure 2.17: This plot shows an example of how the classifier with the highest subjective fitness in different generations (of a particular coevolution run) dynamically change the light intensity to interact with the 2-state agent during a trial.



Turing Learning

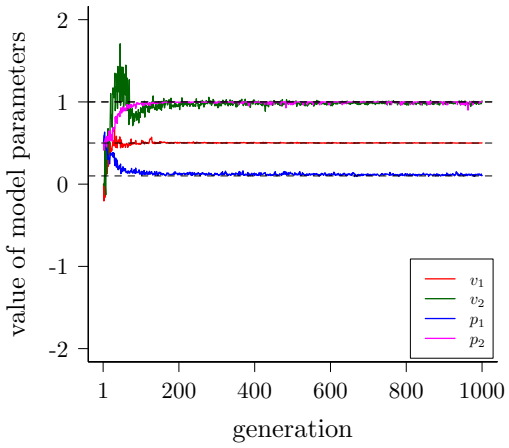


SP-EA

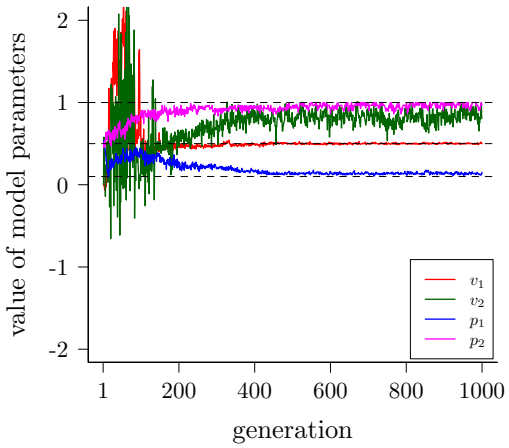


CEA-IM

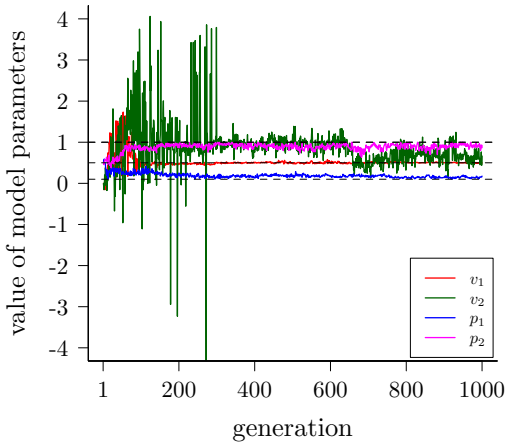
Figure 2.18: This plot shows the distributions of the evolved models with the highest subjective fitness in the 1000$^{\text{th}}$ generation in the coevolutions. Each box corresponds to 30 coevolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). See text for details.



Interactive



Passive 1



Passive 2

Figure 2.19: Evolutionary process of the evolved model parameters for (a) "Interactive", (b) "Passive 1" and (c) "Passive 2" setups of the metric-free method when learning the 3-state agent behavior. Curves represent mean values across 30 coevolution runs. Dotted black lines indicate true values.

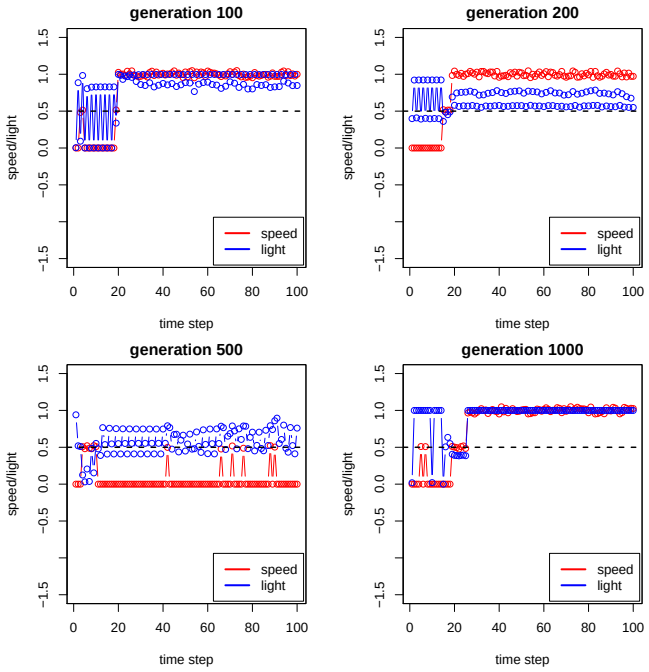


Figure 2.20: This plot shows an example of how the classifier with the highest subjective fitness in different generations (of a particular coevolution run) dynamically change the light intensity to interact with the 3-state agent during a trial.

3 Conclusion

3.1 Summary of Findings

This thesis presented a novel system identification method—*Turing Learning* for inferring agent behavior. *Turing Learning* does not rely on predefined metrics for measuring the difference between the agents and models. Instead, it uses coevolution to simultaneously generate models and classifiers, which substitute the metrics. The classifiers are rewarded for distinguishing between agents and models. The models are rewarded for making classifiers judge them as agents. In other words, in order to 'trick' the classifiers to judge them as agents, the models need to evolve to mimic the behaviors of agents.

The merits of *Turing Learning* were demonstrated through successfully inferring various agent behaviors ranging from swarm behaviors to deterministic/stochastic behaviors of single agents. When inferring an unknown swarm behavior, it is challenging to quantitatively measure the difference (e.g., motion) between the models and agents using predefined metrics, due to the numerous interactions among agents and between agents and the environment. The motion of each agent in the swarm is stochastic. We have shown that through observing the swarm behaviors under investigation, it is sufficient to infer the behavioral rules of the swarming agents. The evolved models show good correspondence to the original agents in terms of individual behaviors (parameters) and global behaviors. The evolved classifiers performed collectively well and could be potentially used for detecting abnormal behaviors (e.g., faulty agents) in the swarm. It was also shown that swarm behaviors can be directly inferred from the motion of a single agent in the group, as long as the group size is sufficiently large. This may have significant implications for the study of animal collectives, as in practice it may be difficult to track a large number of animals in a group.

We extended the ability of the classifiers in *Turing Learning* for inferring deterministic/stochastic behaviors of single agents, so that the classifiers can interact with the agent through controlling the environmental stimulus that the agent responds to. The interactive learning approach proved to be superior to passive learning (i.e., learning where the agents are only observed) in terms of model convergence rate and model accuracy, especially when the agent behavior under investigation had low observability. In the case study about inferring deterministic behaviors of single agents, the results showed that it is possible to infer the behaviors through coevolving a fixed sequence of inputs (in view of classifiers) and models. In this case, the interaction between the classifiers and agent was not very 'intelligent' as they only needed to output a fixed sequence no matter what the agent's observed behavior is.

In a latter case study, *Turing Learning* was applied to infer the stochastic behaviors of a single agent. It was shown that through actively interacting with the agent during the experimental process, the classifiers can 'intelligently' change the stimulus in response to the agent's observed behavior, in order to extract the hidden information from the agent. This intelligent behavior was also observed in humans when scientists try to investigate an animal' behavior in response to stimuli [121]. As the agent's behavior under investigation is stochastic, it is not feasible to evolve a fixed sequence of inputs to extract all the agent's behavioral information. Given the same input, the agent would probably behave differently and this makes it hard to optimize the models using predefined metrics. We compared the results obtained by *Turing Learning* and two other metric-based system identification methods, and showed that *Turing Learning* learned the agent behavior significantly better than the other two methods. This highlights the benefits of *Turing Learning* in inferring agent behaviors.

3.2 Future work

In spite of the encouraging results obtained when applying *Turing Learning* to infer agent behaviors, we do not claim that this method can be directly used for modeling animal behaviors. There are still questions to be discussed before conducting experiments on animals. Some future work is provided as follows.

- In the thesis, the models were represented by a set of parameters that govern behavioral rules of the agents. As it is argued before, this makes it feasible to objectively gauge the quality of the models through comparing them with the ground truth. In the future, we will try to evolve the structure of the models as well (e.g., using genetic programming or artificial neural networks).

- The swarm behaviors investigated in this thesis were deterministic in terms of the individual's behavioral rules. In the future, we could apply *Turing Learning* to learn swarming behavioral rules that are stochastic. In fact, we have shown that if the original controller of the aggregation behavior investigated in Chapter 1 is stochastic, it is still possible to achieve the same global behavior [21]. For the stochastic aggregation controller, in each state, the agent has a probability of switching to another state (i.e., its binary sensor is flipped with certain probability). In this case, *Turing Learning* could be used for inferring both the controller and the probability.

- *Turing Learning* could be applied to learn more complex behaviors (for example, when the agents have more states or rules). When the behaviors become more complex, instead of analyzing only the motion of individual agents, more information (such as number of the agent's neighbors or its internal states) may need to be provided to the classifiers.

- In principle, *Turing Learning* is applicable to infer human behavior. It could evolve models that aim to pass the Turing Test [174], at least with regards to some specific subset of human behavior. In this case, the classifiers could then act as Reverse Turing Tests, which could be applied in situations where a machine needs to distinguish human agents from artificial ones. This is done, for example, by the "Completely Automated Public Turing test to tell Computers and Humans Apart" system (CAPTCHA) [193], which is widely used for internet security purposes. For example, an exciting application of *Turing Learning* is reverse engineering the hand writing of human beings. Given a collection of signatures from a human being, we could coevolve models which are computer programs that automatically generate signatures and classifiers (e.g., neural networks) that distinguish between signatures from the human and those generated by models.

Bibliography

[1] J. P. Grotzinger, "Habitability, taphonomy, and the search for organic carbon on mars," *Science*, vol. 343, no. 6169, pp. 386–387, 2014. [Online]. Available: http://www.sciencemag.org/content/343/6169/386.short

[2] R. P. Hertzberg and A. J. Pope, "High-throughput screening: new technology for the 21st century," *Current Opinion in Chemical Biology*, vol. 4, no. 4, pp. 445 – 451, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367593100001101

[3] J. Bolhuis and L. Giraldeau, *The behavior of animals: mechanisms, function, and evolution*. USA: Wiley-Blackwell, 2004.

[4] W. J. Sutherland, "The importance of behavioural studies in conservation biology," *Animal Behaviour*, vol. 56, no. 4, pp. 801–809, 1998.

[5] D. Floreano and C. Mattiussi, *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. Cambridge, MA: MIT Press, 2008.

[6] 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.

[7] 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

[8] J. Evans and A. Rzhetsky, "Machine science," *Science*, vol. 329, no. 5990, pp. 399–400, 2010. [Online]. Available: http://www.sciencemag.org/content/329/5990/399.short

[9] D. Waltz and B. G. Buchanan, "Automating science," *Sci.*, vol. 324, no. 5923, pp. 43–44, 2009.

[10] L. Ljung, "Perspectives on system identification," *Annu. Reviews in Control*, vol. 34, no. 1, pp. 1–12, 2010.

[11] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. Hoboken, NJ, USA: Wiley, 2013.

[12] S. M. Henson and J. L. Hayward, "The mathematics of animal behavior: An interdisciplinary dialogue," *Notices of the AMS*, vol. 57, no. 10, pp. 1248–1258, 2010.

[13] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," *PNAS*, vol. 99, no. 10, pp. 7280–7287, 2002.

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

[15] ——, "Automated reverse engineering of nonlinear dynamical systems," *PNAS*, vol. 104, no. 24, pp. 9943–9948, 2007.

[16] G. D. Ruxton and G. Beauchamp, "The application of genetic algorithms in behavioural ecology, illustrated with a model of anti-predator vigilance," *Journal of Theoretical Biology*, vol. 250, no. 3, pp. 435–448, 2008.

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

[18] D. Helbing and A. Johansson, "Pedestrian, crowd and evacuation dynamics," in *Extreme Environmental Events*, R. A. Meyers, Ed. Springer, 2011, pp. 697–716.

[19] J. Harvey, K. Merrick, and H. A. Abbass, "Application of chaos measures to a simplified boids flocking model," *Swarm Intell.*, vol. 9, no. 1, pp. 23–41, 2015.

[20] W. S, B. S, F. R, *et al.*, "Modeling collective animal behavior with a cognitive perspective: a methodological framework," *PLoS ONE*, vol. 7, no. 6, 2012, e38588.

[21] 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.

[22] ——, "Clustering objects with robots that do not compute," in *Proc. 2014 Int. Conf. Autonomous Agents and Multi-Agent Syst.*, IFAAMAS Press, Paris, France, 2014, pp. 421–428.

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

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

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

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

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

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

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

[30] 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

[31] 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.

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

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

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

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

[42] 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.

[43] 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

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

[45] D. M. M. and F. D. S., "Beneficial mutationselection balance and the effect of linkage on positive selection," *Genetics*, vol. 176, no. 3, pp. 1759–1798, 2007.

[46] L. S. Russell, "Body temperature of dinosaurs and its relationships to their extinction," *Journal of Paleontology*, vol. 39, no. 3, pp. pp. 497–501, 1965. [Online]. Available: http://www.jstor.org/stable/1301720

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

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

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

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

[51] H.-P.Schwefel, *Evolution and Optimum Seeking*. New York, NY, USA: Wiley, 1995.

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

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

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

[55] 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.

[56] Z. Hong and W. Jian, "A cooperative coevolutionary algorithm with application to job shop scheduling problem," in *Service Operations and Logistics, and Informatics, 2006. SOLI '06. IEEE International Conference on*, June 2006, pp. 746–751.

[57] K. C. Tan, Q. Yu, and J. H. Ang, "A coevolutionary algorithm for rules discovery in data mining," *International Journal of Systems Science*, vol. 37, no. 12, pp. 835–864, 2006.

[58] 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

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

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

[61] 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.

[62] 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

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

[64] 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.

[65] R. Groß, K. Albrecht, W. Kantschik, and W. Banzhaf, "Evolving chess playing programs," in *GECCO 2002*, no. LSRO-CONF-2008-037. Morgan Kaufmann, 2002.

[66] 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.

[67] H. Juille and J. B. Pollack, "Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules," in *University of Wisconsin*. Morgan Kaufmann, 1998, pp. 519–527.

[68] 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.

[69] I. Loshchilov and T. Glasmachers. (2015) Black box optimization competition. [Online]. Available: http://bbcomp.ini.rub.de/

[70] M. Gauci, "Swarm robotic systems with minimal information processing," Ph.D. dissertation, The University of Sheffield, Sheffield, UK, 2014.

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

[72] 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.

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

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

[75] 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.

[76] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. G. Eiben, "Evolutionary robotics: What, why, and where to," *Frontiers in Robotics and AI*, vol. 2, no. 4, 2015. [Online]. Available: http://www.frontiersin.org/evolutionary_robotics/10.3389/frobt.2015.00004/abstract

[77] A. Eiben, "Grand challenges for evolutionary robotics," *Frontiers in Robotics and AI*, vol. 1, no. 4, 2014. [Online]. Available: http://www.frontiersin.org/evolutionary_robotics/10.3389/frobt.2014.00004/full

[78] 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.

[79] 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

[80] 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

[81] A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 345 – 370, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889008001450

[82] 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.

[83] S. Nolfi and D. Floreano, *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[84] 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

[85] 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.

[86] 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

[87] 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.

[88] 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

[89] 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.

[90] 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.

[91] 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.

[92] 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.

[93] 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.

[94] 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

[95] 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.

[96] G. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata, "Autonomous evolution of gaits with the sony quadruped robot," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2. Citeseer, 1999, pp. 1297–1304.

[97] V. Zykov, J. Bongard, and H. Lipson, "Evolving dynamic gaits on a physical robot," in *Proc. 2004 Genetic and Evol. Computation Conf.* ACM Press, Seattle, WA, 2004, pp. 4722–4728.

[98] 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.

[99] 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

[100] 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

[101] A. Eiben, N. Bredeche, M. Hoogendoorn, J. Stradner, J. Timmis, A. Tyrrell, A. Winfield, *et al.*, "The triangle of life: Evolving robots in real-time and real-space," *Advances in Artificial Life, ECAL*, pp. 1056–1063, 2013.

[102] A. Eiben, "In vivo veritas: Towards the evolution of things," in *Parallel Problem Solving from Nature PPSN XIII*, ser. Lecture Notes in Computer Science, T. Bartz-Beielstein, J. Branke, B. Filipi, and J. Smith, Eds. Springer International Publishing, 2014, vol. 8672, pp. 24–39. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10762-2_3

[103] 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

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

[105] J. Bongard and H. Lipson, "Nonlinear system identification using coevolution of models and tests," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, pp. 361–384, 2005.

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

[107] D. Ljungquist and J. G. Balchen, "Recursive prediction error methods for online estimation in nonlinear state-space models," in *Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on*, Dec 1993, pp. 714–719 vol.1.

[108] 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

[109] A. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Berlin, Heidelberg: Springer-Verlag, 2003.

[110] J. Bongard and H. Lipson, "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.

[111] ——, "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.

[112] 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.

[113] 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.

[114] 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.

[115] 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

[116] 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.

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

[118] 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.

[119] 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.

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

[121] 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

[122] 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.

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

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

[125] 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

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

[127] 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

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

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

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

[131] 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.

[132] 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.

[133] M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. Winfield, *Ant Colony Optimization and Swarm Intelligence: 6th International Conference,*

ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings. Springer, 2008, vol. 5217.

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

[135] 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

[136] 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

[137] 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.

[138] 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.

[139] 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.

[140] 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

[141] C. Blum and R. Groß, "Swarm intelligence in optimization and robotics," in *Handbook of Computational Intelligence*, J. Kacprzyk and W. Pedrycz, Eds. Springer, Berlin, Heidelberg, 2015, pp. 1293–1311.

[142] 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.

[143] 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

[144] 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

[145] 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

[146] 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.

[147] 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.

[148] 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.

[149] 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

[150] 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

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

[152] 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

[153] 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

[154] 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.

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

[156] 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

[157] ——, "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

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

[159] 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.

[160] 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.

[161] 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.

[162] 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.

[163] 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.

[164] 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.

[165] 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.

[166] 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.

[167] 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.

[168] 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.

[169] 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

[170] 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

[171] 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.

[172] 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

[173] 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.

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

[175] S. Harnad, "Minds, machines and turing: The indistinguishability of indistinguishables," *J. Logic, Language and Inform.*, vol. 9, no. 4, pp. 425–445, 2000.

[176] J. L. Elman, "Finding structure in time," *Cognitive Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[177] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies - a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[178] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. on Evol. Comput.*, vol. 3, no. 2, pp. 82–102, 1999.

[179] F. Mondada, M. Bonani, X. Raemy, *et al.*, "The e-puck, a robot designed for education in engineering," in *Proc. 9th Conf. on Autonomous Robot Systems and Competitions*, vol. 1. IPCB: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65.

[180] S. Magnenat, M. Waibel, and A. Beyeler, "Enki: The fast 2D robot simulator," http://home.gna.org/enki/, 2011.

[181] R. L. Graham and N. J. A. Sloane, "Penny-packing and two-dimensional codes," *Discrete and Computational Geometry*, vol. 5, no. 1, pp. 1–11, 1990.

[182] P. Levi and S. Kernbach, *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution.* Berlin, Heidelberg: Springer-Verlag, 2010.

[183] B. Eldridge and A. Maciejewski, "Limited bandwidth recognition of collective behaviors in bio-inspired swarms," in *Proc. 2014 Int. Conf. Autonomous Agents and Multi-Agent Syst.* IFAAMAS Press, Paris, France, 5 2014, pp. 405–412.

[184] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library.* Sebastopol, CA: O'Reilly Media, 2008.

[185] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962.

[186] W. Li, M. Gauci, J. Chen, and R. Groß, "Online supplementary material," http://naturalrobotics.group.shef.ac.uk/supp/2014-006/, 2014.

[187] R. D. King, J. Rowland, *et al.*, "The automation of science," *Sci.*, vol. 324, no. 5923, pp. 85–89, 2009.

[188] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Sci.*, vol. 324, no. 5923, pp. 81–85, 2009.

[189] J. Bongard and H. Lipson, "Active coevolutionary learning of deterministic finite automata," *The Journal of Machine Learning Research*, vol. 6, pp. 1651–1678, 2005.

[190] E. Martin, *Macmillan Dictionary of Life Sciences (2nd ed.).* London: Macmillan Press, 1983.

[191] M. Dacke, M. J. Byrne, C. H. Scholtz, and E. J. Warrant, "Lunar orientation in a beetle," *Proc. of the Roy. Soc. of London. Series B: Biological Sci.*, vol. 271, no. 1537, pp. 361–365, 2004.

[192] 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, 2012.

[193] L. Grossman, "Computer literacy tests: Are you human?" June 2008. [Online]. Available: http://www.time.com/time/magazine/article/0,9171,1812084,00.html