Artificial societies and information theory: modelling of sub system formation based on Luhmann's autopoietic theory

Paolo Di Prodi

February 7, 2012

Submitted in fulfillment of the requirements for the degree of Doctor of Philosophy (Ph.D.)

UNIVERSITY
of
GLASGOW

Electronics and Electrical Engineering
School of Engineering
College of Science and Engineering
University of Glasgow

Abstract

This thesis develops a theoretical framework for the generation of artificial societies. In particular it shows how sub-systems emerge when the agents are able to learn and have the ability to communicate.

This novel theoretical framework integrates the autopoietic hypothesis of human societies, formulated originally by the German sociologist Luhmann, with concepts of Shannon's information theory applied to adaptive learning agents.

Simulations were executed using Multi-Agent-Based Modelling (ABM), a relatively new computational modelling paradigm involving the modelling of phenomena as dynamical systems of interacting agents. The thesis in particular, investigates the functions and properties necessary to reproduce the paradigm of society by using the mentioned ABM approach.

Luhmann has proposed that in society subsystems are formed to reduce uncertainty. Subsystems can then be composed by agents with a reduced behavioural complexity. For example in society there are people who produce goods and other who distribute them.

Both the behaviour and communication is learned by the agent and not imposed. The simulated task is to collect food, keep it and eat it until sated. Every agent communicates its energy state to the neighbouring agents. This results in two subsystems whereas agents in the first collect food and in the latter steal food from others. The ratio between the number of agents that belongs to the first system and to the second system, depends on the number of food resources. Simulations are in accordance with Luhmann, who suggested that adaptive agents self-organise by reducing the amount of sensory information or, equivalently, reducing the complexity of the perceived environment from the agent's perspective. Shannon's information theorem is used to assess the performance of the simulated learning agents. A practical measure, based on the concept of Shannon's information flow, is developed and applied to adaptive controllers which use Hebbian learning, input correlation learning (ICO/ISO) and temporal difference learning. The behavioural complexity is measured with a novel information measure, called Predictive Performance, which is able to measure at a subjective level how good an agent is performing a task. This is then used to quantify the social division of tasks in a social group of honest, cooperative food foraging, communicating agents.

I declare that this thesis is a record of the original work carried out solely by myself in the School of Engineering at the University of Glasgow, during the period October 2007 to March 2011. The copyright of this thesis therefore belongs to the author under the terms of the United Kingdom Copyrights acts. Due acknowledgement must always be made of the use of material contained in, or derived from, this thesis. The thesis has not been presented elsewhere in consideration for a higher degree.

Paolo Di Prodi
February 7, 2012

To my family: Carlo Di Prodi, Francesca Viozzi
and Marco Di Prodi.
To my love Pi-Yi Wei.
“One day I will hear the silence from the noise”

Contents

List of Tables

List of Figures

Nomenclature

ABM	Agent Based Modelling
Agent	An autonomous intelligent unit able to cooperate with others and interact with the environment
AI	Anticipatory Information, an input measure based on the Shannon's entropy measure
Alice	A mobile robot produced by EPFL
API	Application Programming Interface
Autopoiesis	A self-reproducing system
Braitenberg Robot	An automated mobile robot with differential driving
CMAC	Cerebellar Model Articulator Controller
CPU	Central Processing Unit
Enki	An open source simulation engine for mobile robots developed by EPFL
GPU	Graphical Processing Unit
ICO	Input Correlation Only learning
ISO	Isotropic Sequence Order learning
Kephera	A mobile robot produced by KTeam
Lego	A company producing modular robotic kits
MAS	Multi Agent System
Maxcorr	An input measure based on the the correlation of thee organism's inputs
MI	Mutual information between two signals
MISO	Multiple Input Multiple Output controller
MLP	Multi Layer Perceptron Network
OpenMP	A standard library for parallel implementations

PCA Principal Component Analysis

PID Proportional Integration Derivative control

Pololu 3π A mobile robot produced by Pololu

POSIX A standard library for parallel implementations

PP Predictive Performance

Q-learning Q-learning is a reinforcement learning technique that works by learning an action-value function to predict future rewards

RBF Radial Basis Functions

RNN Recurrent Neural Networks

SWARM Collective behaviour of animals

Acknowledgements

Firstly, I would like to express the deepest gratitude to my supervisor Dr. Bernd Porr for giving me the opportunity to do a PhD in the field of computational neuroscience and artificial intelligence. Secondly, I would like to thank Maria Thompson for being a a science colleague with good critical thinking and Lynsey A. McCabe, the warmest Scottish person I have ever met in Glasgow. I have enjoyed the company of many other PhD students who shared the good and bad turns of my life: a nice hug to Colin Waddel, Mohammad AlRawani and the like.

A special mention to the laboratory technician, Tom O'Hara who taught me lots of things including how to play golf.

I hope the new rookies in my office will have an exciting career and live up to the standards of our group.

The financial support for this research project, given by the Scottish Funding Council in the name of Glasgow Research Partnership in Engineering (GRPE) is gratefully acknowledged.

I would also like to thank the Scottish Enterprise, the Scottish Gateway, the PSBYT and the student Enterprise, in particular Greig Sinclair, for their support in taking forward my business idea and giving me the opportunity to learn more about entrepreneurship.

Finally a big thanks to my girlfriend Pi Yi Wei who supported me in endless sessions of late night writing and horrible time tables.

Publications

This thesis describes the work I have done on the theory of social systems which produced the following publications:

- Di Prodi, P. and Porr, B. and Wörgötter, F. (2010) A Novel Information Measure for Predictive Learning in a Social System Setting. From Animals to Animats 11, Lecture Notes in Computer Science, editors: Doncieux, Stphane and Girard, Benot and Guillot, Agns and Hallam, John and Meyer, Jean-Arcady and Mouret, Jean-Baptiste, Springer Berlin / Heidelberg, pages 511-522, vol, 6226.

- Di Prodi, P. Porr, B. Wörgötter, F. (2008) Adaptive Communication Promotes Subsystem Formation in a Multi Agent System with Limited Resources. In: Learning and Adaptive Behaviors for Robotic Systems, 2008. LAB-RS '08. ECSIS Symposium, pages 86-96.

And the following abstracts:

- Di Prodi P, Porr B and Wörgötter F (2009). A novel information measure to understand differentiation in social systems. Frontiers in Computational Neuroscience. Conference Abstract: Bernstein Conference on Computational Neuroscience.

- Kulvicius T, Kolodziejski C, Prodi P, Tamosiunaite M, Porr B and Wörgötter F (2008). On the analysis and evaluation of closed loop learning systems. Frontiers in Computational Neuroscience. Conference Abstract: Bernstein Symposium 2008.

Additionally, I have done some research work in collaboration, with previous PhD student Lynsey McCabe, in the field of computational neuroscience which produced the following publications:

- Bernd Porr, Lynsey McCabe, Paolo Di Prodi, Christoph Kolodziejski, Florentin Wörgötter, How feedback inhibition shapes spike-timing-dependent plasticity and its implications for recent Schizophrenia models, Neural Networks, In Press, Corrected Proof, Available online 10 March 2011, ISSN 0893-6080.

- McCabe L., Porr B., Di Prodi P. & Wörgötter F. (2008) Observing STDP of pyramidal cell and attached interneuron microcircuit using detailed CA2+ dynamics, Fens Forum 2008, Poster session.

- McCabe, L., Di Prodi, P., Porr, B. and Wörgötter, F. (2007) Shaping of STDP curve by interneuron and Ca2+ dynamics. Proceedings of the sixteenth annual computational neuroscience meeting CNS*2007, Toronto, Poster session.

I have also done some work on the application of machine user interfaces for health services which produced the following publications however they are not relevant to my PhD:

- Di Prodi P, Power CF and Wei PY (2009). Extending the reach of Mental Health Services through eLearning technology and other communication mediums centralized on one Online ePlatform. Frontiers in Neuroengineering. Conference Abstract: Annual CyberTherapy and CyberPsychology 2009 conference.

- Power CF, Di Prodi P. Extending the reach of Mental Health Services through eLearning technology and other communication mediums centralized on one Online ePlatform. ISBE 2008.

Chapter 1

Introduction

1.1 A theory of social systems

The German sociologist Luhmann proposed, in his seminal work (Luhmann, 1984) a new theory about how communications generate societies. Important to his work is the underlying assumption that organisms act as closed loop systems (Wiener, 1961; von Glasersfeld, 1995) who pursue their own goals and are not able to observe the internal states nor the inputs of other organisms (Foerster, 1960). A central assumption is that agents are continuously trying to reduce their own perceived uncertainty of their environment (perceived complexity from the agent's point of view). This is done by learning to anticipate events in the environment. For example an agent can learn to use vision to prevent falling off a cliff (Verschure and Coolen, 1991). In other words agents aim to turn themselves from reactive into proactive closed loop systems. In the social context this becomes more complex when learning agents try to predict each other. Because agents cannot observe the internal states of the other agents, the system becomes more unpredictable. Parsons called this the double contingency problem (Parsons, 1977): Ego is trying to predict Alter, but Alter does the same. In the case of two agents the double contingency problem might still be treatable. However, when there are more than two agents the uncertainty grows. In order to reduce the uncertainty Luhmann proposed that social systems have to create subsystems which specialize (Luhmann, 1984) in the sense that they form sub-groups by executing only a subset of behaviours and/or agents only perceive certain aspects of the environment and not all. A fundamental condition for the formation of sub-systems is communication, which Luhmann uses in a broader sense compared to the concept of language. There have been attempts to model aspects of social system theory, which will be discussed in Section 2.1. The first model which incorporated Luhmann's principles belongs to Dittrich et al. (2003) which used double contingency as the origins of social order. Luhmann's communication is the distinction between information, transmission and understanding, which is required if the agents operate as autonomous close loop systems. Also agents show meaningful motivated behaviour towards others, according to goals and a shared symbolic system, as proposed in Parson's models (Parsons, 1951, 1977). The next section describes a very important modelling approach particularly suited for social systems.

1.2 Modelling approach ABM

Agent Based Modelling is a powerful tool where the researcher has to simulate a complex system like a social system (W.Macy and Willer, 2002). A social system is composed of an agent or entities which interact with each other and the environment. Every entity is described by a set of behaviours or rules which can be static or dynamic. The interaction can be at the action level or at the communication level. Traditional approaches are based on logic of formal analysis) or on mathematical models of differential and partial equations. The formal analysis - also known as model checking- can be used to verify if a property of the system is valid or not, without the need of applying statistic on a large set of simulations. Model checking, if properly used, can be a powerful tool for the analysis of social models and was also applied in parallel to the ABM model (see the Conclusion for a more detailed explanation). The approach with differential equations is only feasible when the model can be described analytically and has been very effective to model the dynamic of biological populations like the well famous Lotke Volterra equations (Volterra, 1931) that describes the evolution of predator-prey populations The differential equations are formulated over the general behaviour of the system and do not take into account individual interactions that are the main purpose of research in this study. The model becomes even more complicated when each agent has an adaptive behaviour which changes in time with the others: this means that every agent is initially identical but as time progresses and interactions are performed, dissimilarities emerge and thus a new collective behaviour emerges.

The ABM approach was chosen not only for the with the aim of implementing the simulated system in a real robotic hardware. Thanks to the ABM approach an agent that has been embodied in a software simulation, can be easily embodied in a real robot. The advantage is then not only the time required to realise such a system, but also the expected behaviour of the robot that will match the simulated behaviour. There are a numbers of potential software frameworks that can be used to implement ABM systems and these will be discussed in Section 2.4.

1.3 Objectives and Motivation of the thesis

The motivation behind this research is the implementation of Luhmann's principles for the generation of artificial social systems. Luhmann formulated a theory that has produced a considerable impact in sociology but there have only been a few attempts to validate the theory experimentally. This is what has motivated this thesis in terms of experiment setup and validation of the model. The validation of the model, even to a limited extent of sub-properties of the original theory, is an important step for the understanding of not only artificial societies but also of human like processes. It could help in the future to predict the effect of a policy (normative order) on a human population or to build robots which are able to cooperate and interact socially between themselves and human operators. This is why the modelling approach used in this work is based on biological inspired behaviour which captures the natural behaviour of simple animals and can easily be transferred into an embodied robotic system. A conceptual diagram of the thesis and of the work is shown in Figure 1.1 and Figure 1.2.

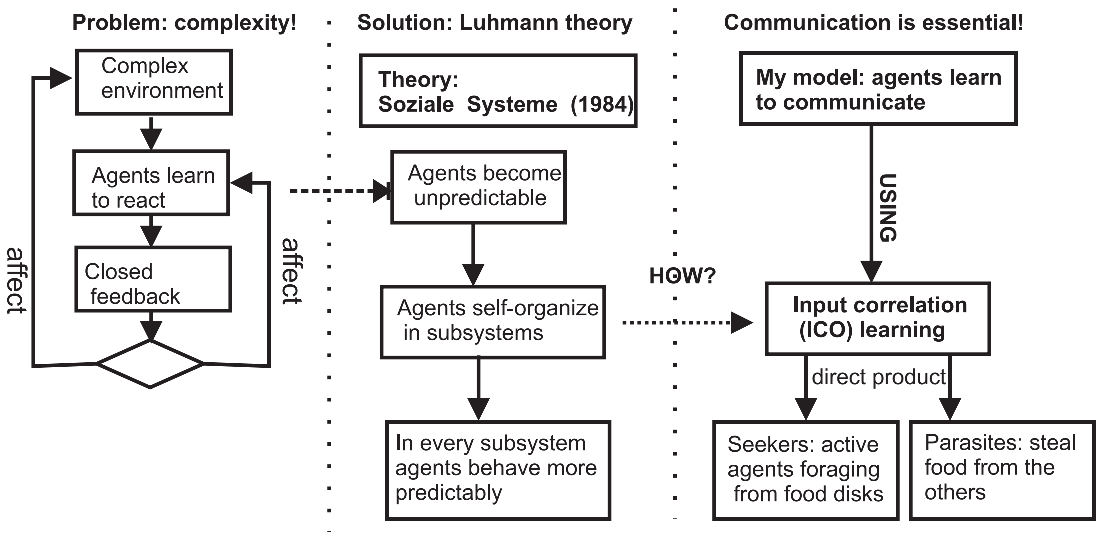

Problem: complexity!
Complex environment
Agents learn to react
Closed feedback
affect
affect
Solution: Luhmann theory
Theory: Soziale Systeme (1984)
Agents become unpredictable
Agents self-organize in subsystems
In every subsystem agents behave more predictably
HOW?
Communication is essential!
My model: agents learn to communicate
USING
Input correlation (ICO) learning
direct product
Seekers: active agents foraging from food disks
Parasites: steal food from the others

Figure 1.1: Thesis objective at the system level

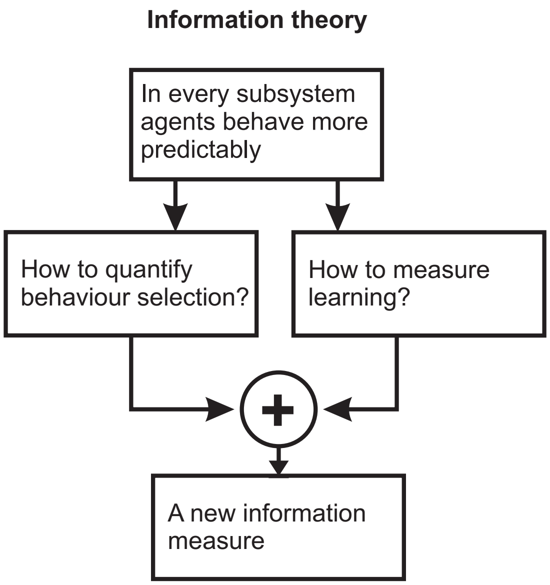

Information theory
In every subsystem agents behave more predictably
How to quantify behaviour selection?
How to measure learning?
+
A new information measure

Figure 1.2: Thesis objective at the individual level

1.4 Aim

The aim of the thesis is to investigate the formation of sub-systems at a system level as summarised in Figure 1.1 and to investigate the behaviour at an individual level as summarised in Figure 1.2.

The objectives of the thesis can be then summarised:

- implement a social system based on the Luhmann's hypothesis
- verify the self-organising property (autopoiesis principle) of the system
- measure the self-organising property of the system
- quantify the self-organising property at the individual level with an information theoretic approach
- apply the same approach to different models

1.5 Outline of thesis

The thesis is divided into the following chapters:

- this Chapter is a general introduction to this thesis and a quick review of the existing literature.
- Chapter 2 is a detailed review of literature closely related to this thesis.
- Chapter 3 contains the main research and results carried out by the author during his Ph.D.
- Chapter 4 contains a summary of the results, a critical comparison with the existing literature, a description of future work and possible or existing industrial applications.

In Chapter 3, each Section was written to be a self-contained module structured in the familiar order: Introduction, Methods, Results, and Discussion. To avoid too much fragmentation a simple notation was used, for example in Chapter 3, Section 3.4 is divided as follows:

- Introduction: the application of information flow to Q-learning
- Methods: reinforcement learning
- Methods: Q-Learning algorithm
- Methods: Q-Learning connectionist
- Methods: The robot and the task
- Results: avoidance case
- Discussion

Each entry is then prefixed with the corresponding order. All the sections are ordered in a logical manner, which does not correspond directly to the chronological order of the research. This happened for example with the Predictive Performance measure, which was developed for a retinal robot and only after was applied to the data generated for the social system. I have chosen a logical order so that the reader will be introduced gradually to the topics.

Chapter 2

Background literature

2.1 Introduction to the theory of Societies

This section contains an introduction to the theories about the generation of human like societies and existing models in literature which replicate artificial societies.

2.2 A brief history of sociology

Sociology is the study of society and aims to understand how social order is possible. In the last 350 years, sociologists have provided different explanations to the generation of social order:

- Thomas (1885) attributed the generation to a powerful state, the Leviathan
- Smith (1776) attributed the generation to an "invisible hand"
- Durkheim (1893) introduced the concept of norms
- Parsons (1937) extended the Durkheim proposal by saying that norms are legitimated by values located in a cultural system of a society
- Axelrod (1984) suggested that the generation is possible by rational choice of action with consideration for a long common future (shadow)

A cardinal point in the theory of social order generation was introduced by Parson as the *the problem of double contingency*:

> There are two crucial reference points for analysing interactions: (1) That each actor is both acting agent and object of orientation both to himself and to the others; and (2) that, as an acting agent orients to himself and to others, in all primary modes of aspect. The actor is knower and object of cognition, utiliser of instrumental means and himself a means, emotionally attached to others and an object of attachment, evaluator and object of evaluation, interpreter of symbols and himself a symbol. (International Encyclopedia of the Social Sciences,1968: 436)

Following Talcott (1967), Luhmann (1995) identified *the problem of double contingency* as the main problem of producing social order whilst expanding the idea with a new radical biological principle called autopoiesis.

2.2.1 Autopoiesis: from biology to social systems

Luhmann based his social system theory on the concept of "autopoiesis" , originally formulated in biology by the two biologists Varela and Maturana (1980). The biological concept of autopoiesis (from the Greek autos=self, poiein= to produce) states that a living system recursively reproduces its elements through its own elements. A living cell, for example, reproduces its own elements, like proteins or and a plant grows its own leaves and roots. The autopoietic system has 3 important properties and is described in Figure 2.2):

1. operative closure: no operations can enter nor leave the system within its boundary
2. interactional openness: the system has contact with its environment by means of disturbances
3. structural coupling: environmental events can trigger internal processes but the internal processes triggered are determined by the structures of the system

The operative closure is symbolised by the partial feedback of the output to the input of the cell. In essence autopoietic systems are at the same time open and closed systems; open because they are influenced by their environment, but also closed because environment does not directly influence the structure and elementary processes of the systems.

The first and second property are also the basis of cognition:

> Living systems are cognitive systems, and living as process is a process of cognition. (Maturana and Varela 1980:13)

The operations of an autopoietic system are defined as its cognitions: cognition is a self-referential, autopoietic process. This assumption is known as *Radical Constructivism*: all ideas are constructs of the cognitive system and are a by-product of reality. The most important contribution to the development of constructivism to neuro biological systems was pioneered by Von Foerster (2003). Every organism lives in a closed loop with its own environment and works only with neural activity. The Figure 2.1 shows how the cognitive system produces neural activity and perceives neural activity: the environment is constructed as part of the feedback loop. This self-reference property also exists at other layers, for instance in a cell proteins create other proteins but the environment is coded as the protein production to maintain a membrane (made again by proteins). This assumption is also important in the work of this thesis because artificial agents, as well as human beings, construct their own reality. For instance a common behaviour between animals is the natural reflex reaction to pain: when the skin is touching a hot surface, the nervous system produces the idea of pain or heat. The molecular property of the flame, triggered an action potential in the pain receptor of the skin that then sent a reaction command in the motor cortex. The physical event did not enter the cognitive system but only generated a disturbance from the plateau state of the nervous system.

Later, in Section 3.1.2 there is a clear explanation of the implication of such an assumption.

The third property is the concept of self-organisation: the autopoietic system replicates its elements by following a structure which is self-determined. Thus we can state that autopoiesis

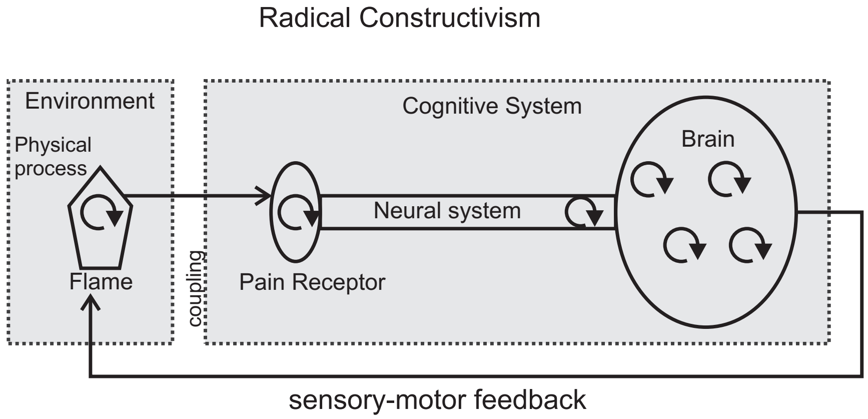


Figure 2.1: Radical constructivism example: a burning flame is a physical system, regulated by an oxygen reaction, when a finger touches it, a coupling is established between the neural system and the flame. The pain receptor produces neural activity, the brain generates a motor reaction, which then evokes an absence of sensory pain indicating that the reaction was good. The sensor produces again a different neural activity to encode the absence of pain.

refers to the reproduction of the elements and self-organisation refers to the determination of structures (Luhmann, 2000).

Luhmann (2000) generalises the principle of autopoiesis to be a general form of system building thus declaring that a system is autopoietic when it reproduces its own elements. Figure 2.3 describes the hierarchy of autopoietic systems:

1. level 1 contains the general definition

2. level 2 contains living systems, psychic systems, neural systems and social systems

3. level 3 contains societies, organisations and interactions

Each system is described by the unit self-reproducible elements, in particular neural systems reproduces their own neural activity, living systems their own proteins and societies their own communications. In this thesis I am going to focus on the study of Societies (Luhmann, 1995) which reproduce themselves on the basis of communication and on the study of Neural Systems. Moreover, Luhmann has also investigated the formation of organisations (Luhmann, 2000) and social interactions (Luhmann, 1993).

The next sections contains a descriptions of the neural systems that constitutes the core of our artificial agents or robots.

2.2.2 Neural systems

The fundamental component of almost every living organism is its nervous system which is necessary for the the most vital activities like digestion, reproduction to the most complex ones like locomotion and cognition. The most important assumption was made by Von Foerster

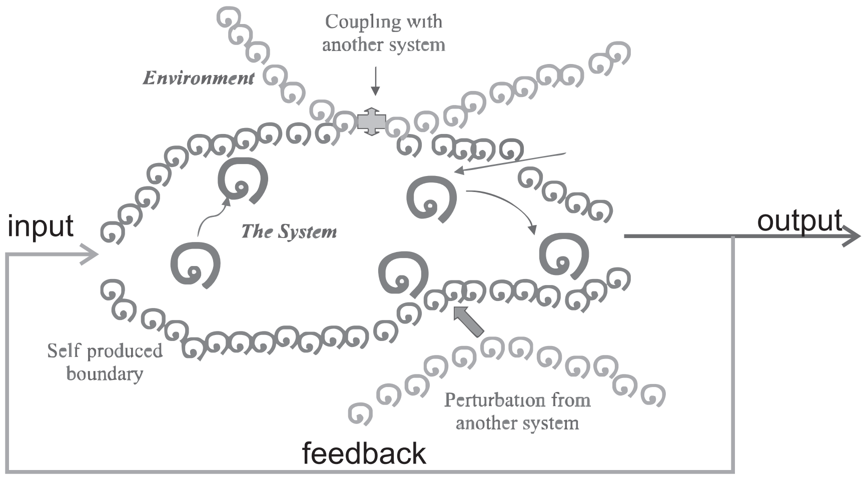


Figure 2.2: The original definition given by Maturana and Varela: An autopoietic machine is a machine organized (defined as a unity) as a network of processes of production (transformation and destruction) of components which: (a) through their interactions and transformations continuously regenerate and realize the network of processes (relations) that produced them; and (b) constitute it (the machine) as a concrete unity in space in which they (the components) exist by specifying the topological domain of its realization as such a network. [. . .] the space defined by an autopoietic system is self-contained and cannot be described by using dimensions that define another space. When we refer to our interactions with a concrete autopoietic system, however, we project this system on the space of our manipulations and make a description of this projection. From Varela and Maturana (1980) reproduced with permission of the publisher.

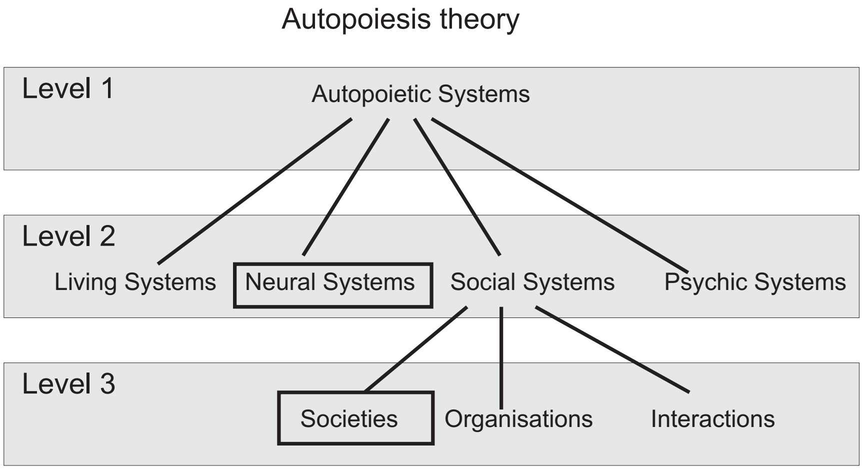


Figure 2.3: Autopoiesis becomes a general concept applied at different layers of system formation. This thesis is focusing on studying the Social Systems.

(1985), one of the founders of radical constructivism, who argued that the nervous systems is operationally closed: neural activity generates other neural activity and thus the environment is only a "simulacra". The most basic type of a neural system is the reactive system, in control theory known also as a feedback system, which only reacts after a sensory event has occurred. A feedback system can also be called a closed loop or self-referential system, but I am going to use the most popular notation of closed loop system from now on.

The block diagram in Figure 2.4 represents a minimal system with feedback. Each block contains the Laplace transform of the time domain function transfer:

- $H_0(s)$ is the transfer function of the agent
- $P_0(s)$ is the transfer function of the environment

The transfer functions operates on neural signals which are the basic elements of the neurological system. Because the systems is in closed loop form the following equations are valid:

$$V(s) = H_0(s) \cdot X_0(s)$$
$$X_0(s) = P_0(s) \cdot V(s)$$

Such a closed loop system can be stable or unstable and (Von Foerster, 1985) assumes, like in traditional control theory, that it must operate in the stable state by using a negative feedback. Stability is necessary in a linear system if the organism wants to reach a desired state after a finite time. The main issue with this closed loop model is that the organism cannot distinguish itself from the environment because by dividing both the transfer functions by P_0, the new transfer function $G(s) = H_0/P_0$ (see Figure 2.4, bottom) does not distinguish any more from organism and environment due to the unity gain feedback. To avoid this issue, a disturbance D must be defined as in Figure 2.5. The disturbance is anything which prevents the organism

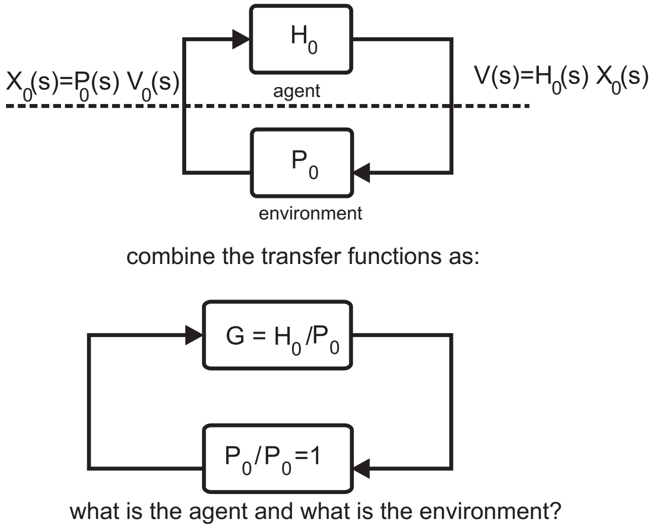

H0
agent
X0(s)=P0(s) V0(s)
V(s)=H0(s) X0(s)
P0
environment
combine the transfer functions as:
G = H0/P0
P0/P0=1
what is the agent and what is the environment?

Figure 2.4: A simple closed loop system. Top: The transfer function H_0 transforms sensor signals X_0 into motor signals V. The transfer function P_0 transforms the motor signals V back into sensor signals X_0. Bottom: the system can be simplified by dividing everything by P_0

to keep its stable desired state and was also formulated by (Ashby, 1956) in its law of requisite variety which is going to be discussed in Section 3.3.

The disturbance this time cannot be eliminated by dividing the transfer functions by P_0. A simple reactive system then can only react after a change it the desired state, a simple example is the motor reaction of our hand to a painful event like touching a flame. Another important point made by the radical constructivism theory is that only actions which feed back to the organism's sensors can be observed by the organism. In the mentioned example then even a more complex motor reaction will only generate two possible state at the sensory input, one of pain and one of non pain. Any other action which simply disappears in the environment cannot be observed by the organism. Thus, there is no other chance for the organism as to analyse its inputs as this is the only aspect that the organism is able to observe. Even its own actions are only observable through its inputs.

A more advanced organism is one which has an anticipatory input signal X_1 as shown in Figure 2.6: when the agent is born only the inner loop H_0, P_0 is active but by using the association between X_0 and X_1 the agent can in the future avoid the painful signal by only using the anticipatory information X_1. The reflex signal X_0 represents the initial behavioural goal (Verschure and Voegtlin, 1998) where $X_0 = 0$, while the predictive signal X_1 is provided naturally by the environment or the organism's sensor setup.

The desired state, for example $X_0 = 0$ cannot be maintained all the time as disturbances D arrive at the loop occasionally. These disturbances enter the inner loop delayed by time T. The undelayed disturbances enter the organism via the sensor input X_1 which anticipates the input X_0 The signal at X_1 can now be used to observe the primary feedback loop (X_0,P_0) and determine what is the effect of on the primary feedback loop. This observation can be used

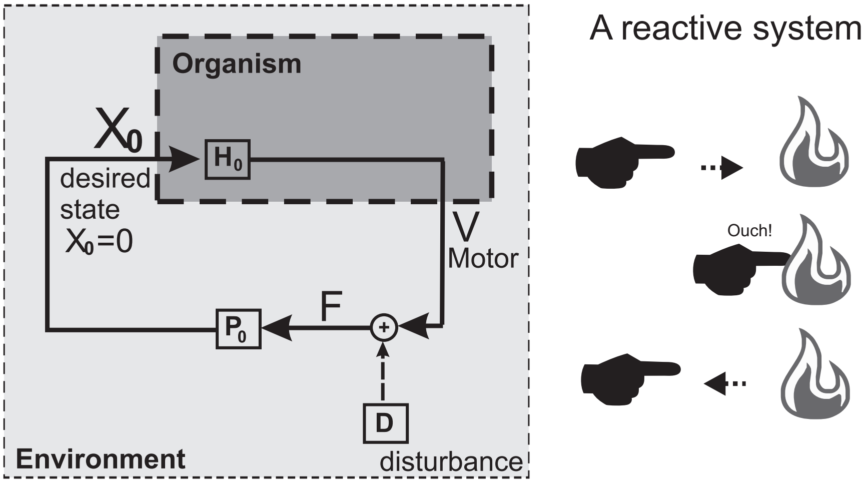


Figure 2.5: The closed loop system now contains a disturbance from the environment. An example is the classical natural reaction to a burning sensation: the hand retracts when the pain receptor is activated by the contact with the flame. Original figure in Porr et al. (2006).

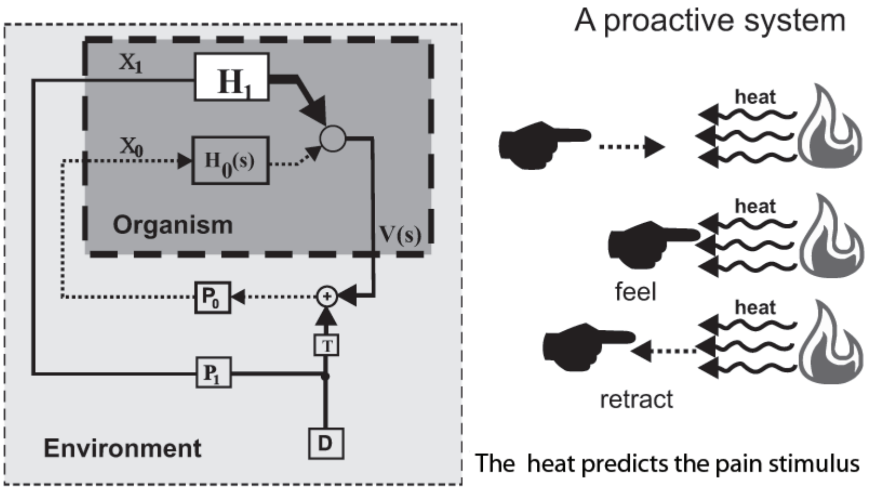


Figure 2.6: The inner feedback loop is established by the transfer functions H_0 and P_0. The outer feedback loop is established by the transfer functions H_1, P_{01} and P_1. D is the disturbance and T delays the disturbance. The outer feedback loop observes the inner feedback loop and adjusts so that the inner reflex loop is no longer needed. Figure used with permisison from Porr et al. (2006).

to adjust H_1 in a way that the inner feedback loop does not feel the disturbance any more. This approach was succesfully implemented by Porr and Wörgötter (2003) where a robot was using a camera system to learn the avoidance of obstacles. The camera provided the visual anticipatory information required to predict the triggering of the touch avoidance sensor. The robot used a learning statregy called ISO learning which uses the correlation between vision and touch signals to learn the anticipatory reaction. A more advanced form of ISO learning called ICO learning is used in this thesis as the main controller both for the software agent and the embodied robot.

The most interesting implication of this model when using multiple agents is the double contingency whereby each agent is disturbing the other as shown in Figure 2.7. Each agent is nested in multiple closed loops, because it receives the outputs of all the other agents in the environment via the disturbance summation with the environment itself. Double contigency happens when all agents are learning continuously from each other. This learning loop was defined by Luhmann as the problem of double contigency: an open ended interaction process where each agent try to predict the other. First, contingency arises because agents are complex systems that are "black-boxes“ for each other. An agent will never know exactly what the other will do next. A good metaphor is the game of chess where each player try to anticipate its opponent moves in the future: in a way the chess game is a sort of communicative process between the two players which eventually comes to an end when the desired state (check mate) is achieved. This dynamic process constitutes for Luhmann the necessary base for social system generation: the problem of double contingency is solved by mutual expectations. It is possible also that learning will never stabilise due to such multiple nested closed loops.

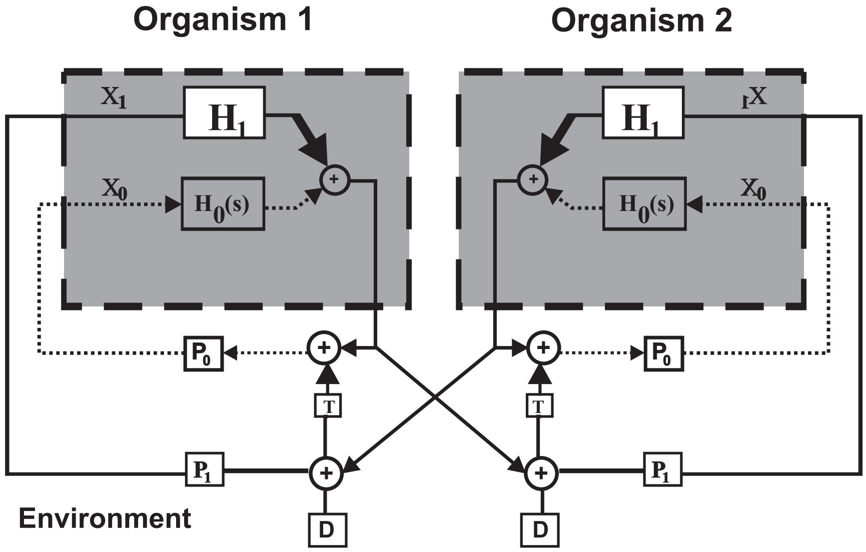


Figure 2.7: Each organism output is summed to the motor feedback loop with the disturbance from the environment.

2.2.3 Breitenberg vehicles

A good controller is useless if it cannot be embodied in a physical robot. The ICO and ISO controller were successfully implemented on a moving robot by adopting the approach developed by Braitenberg (1984) who devised a simple yet effective method to implement avoidance and attraction behaviours on simple robots. The most basic architecture is composed by a couple of sensory inputs, a couple of motors and a matrix of connectivity which defines the behaviour of the robot. The test case scenario is a vehicle moving on a planar surface which contains a light source be projected from above. The vehicle has 2 light sensor which are able to measure the scalar gradient of the light source on the surface. The motors are directly connected to the sensors via positive or negative proportional controllers. Figure 2.8 shows that an avoidance behaviour can be implemented by using positive feedback connections:

- a vehicle with symmetric connections from sensors to motors, dislikes the source of light by constantly avoiding it.
- a vehicle with crossed connections from sensors to motors, dislikes the source of light but constantly turns toward it at maximum speed, as if it wanted to destroy it.

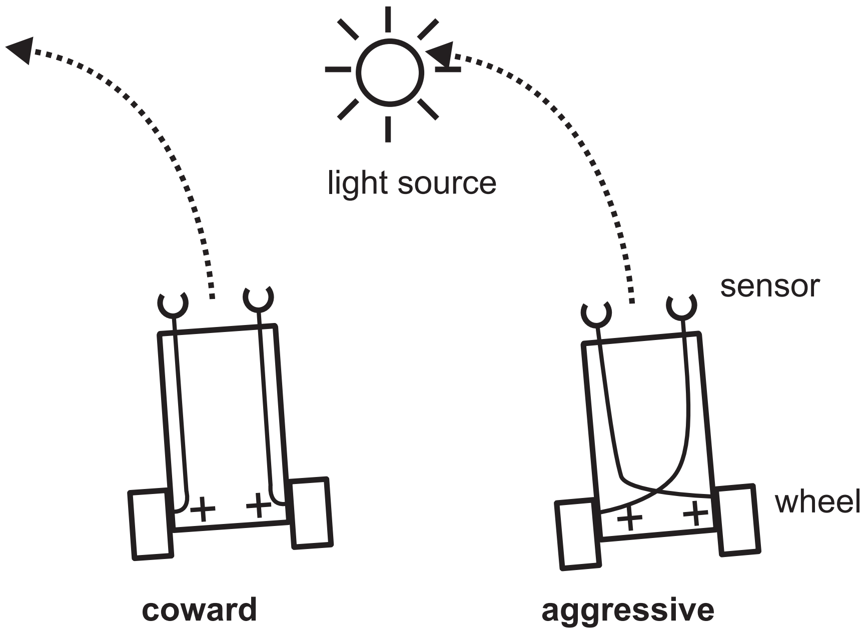


Figure 2.8: An example of a coward and aggressive vehicle with only positive feedback connections.

Figure 2.9 shows that an attraction behaviour can be implemented by using negative feedback connections:

- a vehicle with symmetric connections from sensors to motors, likes the source of light by carefully approaching it and reaching ideally zero velocity in proximity.
- a vehicle with crossed connections from sensors to motors, likes the source of light but constantly search for other alternatives

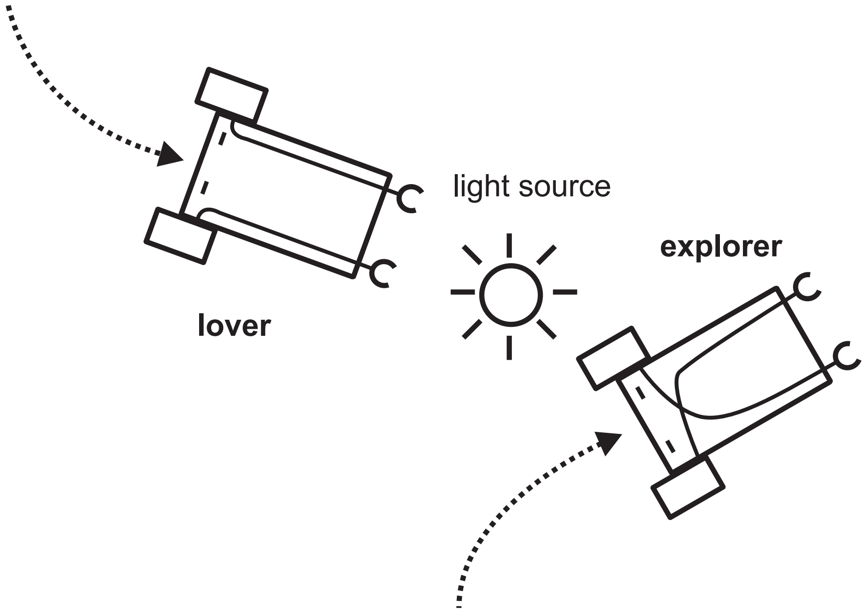


Figure 2.9: An example of a lover and explorer vehicle with only negative feedback connections.

The behaviour of such vehicles can be enriched by including more sensors and so mixing positive and negative connections or by using non monotonic relationships between sensors and motors. The agents that I used in the software simulations and robot implementations, are based on the Braitenberg framework but with a minor difference in the implementation: instead of having a direct connection between inputs and motors, an error signal is generated between the left and right synapse which then generates the differential motor command for the left and right wheel viat the ICO learning algorithm.

2.2.4 Communication

For Luhmann, the most important feature of a social system is that communication is the basic form of the autopoietic reproduction of social systems. Social systems consist of communicative processes (see Figure 2.10), not human beings.

Human beings are individuals in a society only when they communicate and thus the limit of society is defined by the limit of communication. Communication has been defined by Luhmann as a unity of 3 selective and independent processes:

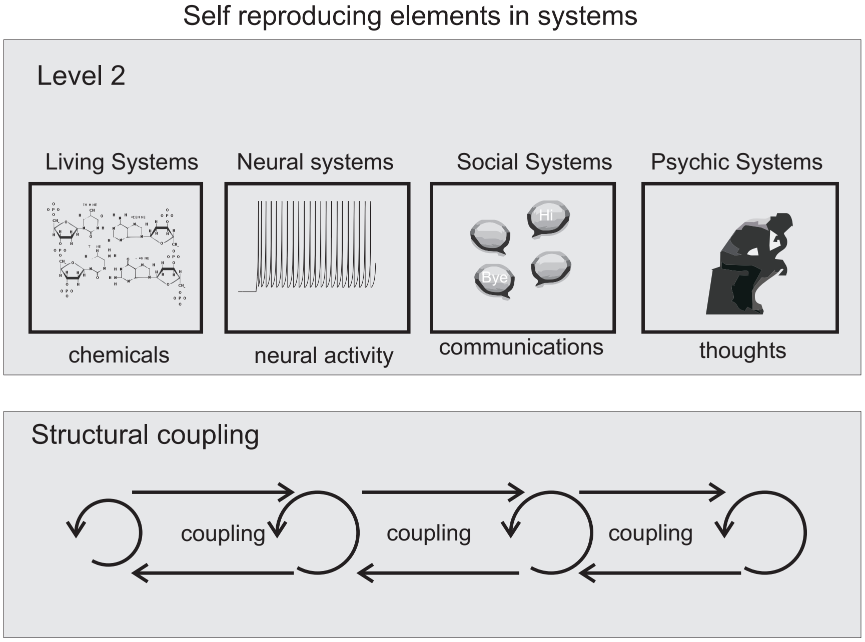


Figure 2.10: Living systems are based on cellular mechanisms of reproduction and their elements are proteins and molecules. Social systems are based on communication processes which reproduces themselves. Psychic systems are based on self-reproducing thoughts. Each system is structurally coupled with each other. The structural coupling between psychic systems and communications is established through Language. Language allows the ”synchronization“ between the two systems but communication is also possible without it.

- utterance: how do we utter it?
- understanding: how do we separate utterance from information?
- information: what was the utterance about?

Communication is an **emergent property** of the interaction between two psychic systems as the three selections do not belong to each individual independently. Each step is considered by Luhmann as a selection process from a set of possibilities, in accordance with the definition of information of Shannon and Weaver (Shannon and Weaver, 1949). To explain the communication process Figure 2.11 contains an example of a communication session between Ego and Alter. The psychic system embodied in Ego has the intention or desired state of being alone because he/she is tired. The first step for Ego is to select the appropriate utterance which in this case is the English sentence "go away". At this point Alter receives the utterance and needs to understand it by generating the information from it. Alter then can decide to leave or to continue the conversation. Assuming that in the first case, he leaves Ego alone and emits an utterance like "Oky", Ego will understand that its initial utterance "go away" was successful. On the contrary if Alter persist and does not leave Ego alone, Ego will try indefinitely to produce other utterances until Alter leaves.

In Figure 2.11 Ego can also accept or reject the meaning of the communication which implies a dynamic selection mechanism for the continuation or interruption of the communication process.

The Luhmann interpretation of communication is therefore far more advanced than the simple channel communication theory where the purpose is to transfer meaning with minimal errors. In this sense there is similarity with the work of (Berger and Calabrese, 1975) who developed an axiomatic theory to explain people's attempts to make sense of interpersonal situations by reducing uncertainty through seeking information. Additionally the concept of reducing uncertainty during the understanding process being a motivation to continue a communication is not mentioned in the original work. There is also another interesting implication of using Luhmann's communication which fits recent theories of the mind. The use of expectations is tightly linked with the idea of estimating each other's state of mind as described with greater detail in Section 4.1.7 and 4.1.3. So in summary, communications can reproduce themselves but what communications are produced is related to the concept of expectation. In the next section I am going to introduce the most relevant simulation model which captures the social ordered generation by using such a model of communication based on expectations.

2.3 Social order generation by double contingency

The starting point for implementing a simulation which uses Luhmann's communication approach, was developed by Dittrich et al. (2003) which implemented the situation of double contingency as the origin of social order. The results produced by this approach were very promising.

2.3.1 Methods

He started from a dyadic social interaction and then expanded it to a multi agent interaction. Social order appears in the dyadic social interaction and also in the multi agent situation, but

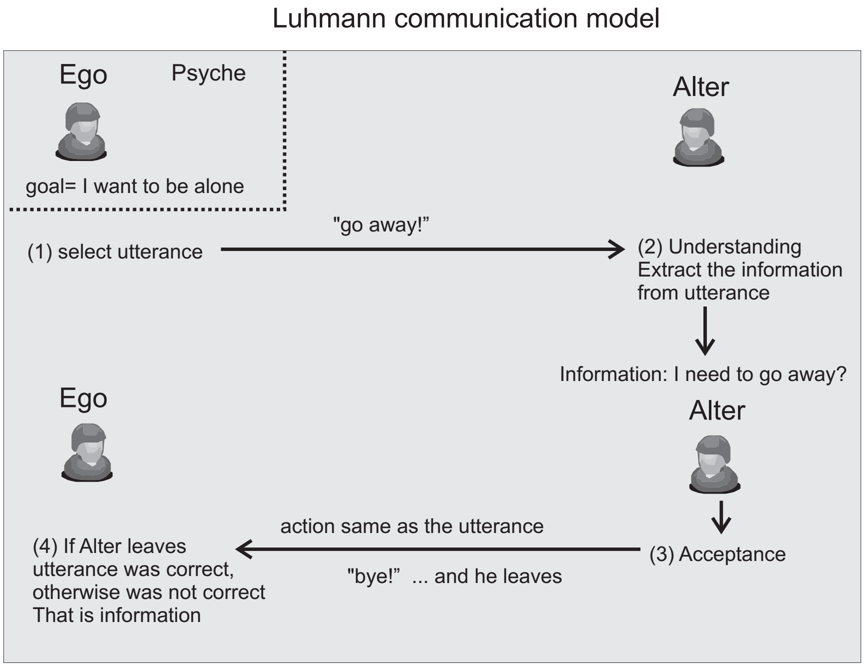


Figure 2.11: Ego is a psychic system with a current goal or desired state. When Ego is in contact with Alter in a contingent situation. he selects the utterance ”go away“ to be communicated to Alter. Alter then needs to extract the meaning from such an utterance and infers that Ego wants to be alone. Then Alter can decide to continue in the conversation or quit it by just leaving Ego alone. The acceptance or rejection of the understanding is an option in the communication.

only in certain conditions that I am going to explain.

The double contingency problem exists when a dyad composed of 2 entities (ego and alter) meet each other: each actor has a double role of knower and object of cognition.

Parsons solved the problem of contingency by asserting that a common shared symbol system is a pre-condition for the formation of a social order. Therefore the dyad must share a culture derived from a history of previous relationships.

Luhmann solves the problem of double contingency by using a self-organisation process which develops in time based on mutual expectations. The only hypothesis he made was that alter and ego (the actors in the dyad) have a necessity of predicting “expectation-certainty”, which means Alter and Ego want to know what is going on in this interaction. Every entity expects that the other entity has expectations about its next activity.

The desire -or goal- of every agent during the interaction is to reduce the entropy -uncertainty- of the Alter’s actions given Ego’s actions.

To give a simple example, when I say “Hello” to a friend, I expect to receive a “Hello“ followed by a form of embrace, typically a handshake possibly followed by a conversation. My friend will expect the same. This is useful as we don’t have to try out all our vocabulary every time to get somebody’s attention! Figure 2.12 explains the model with a simple example where Ego says ”Good morning“ and alter replies with ”Good morning“, then Ego asks ”How are you?” but Alter ask a question about time. This somehow puzzles Ego -delusion- because he wasn’t expecting a question after his question.

So the model proposed by Dittrich begins with a simple dyad of 2 actors:

- Ego initializes the interaction by choosing a message from a set of N possible ones
- Alter receives the message and replies with another one from the set of N
- the conversation continues from Ego and so on

The activity of an actor is to decide which message has to be sent given a received message. Each agent is motivated by 2 functions:

- Expectation-Expectation **EE**: an agent wants to meet the expectations of the other agent. It does so by keeping a memory of what actions were chosen in response to other agents.
- Expectation-Certainty **EC**: the reaction of the other agent following its own activity should be as predictable as possible.
- Activity Value: a linear combination of $(1 - \alpha)EE + \alpha EC$.
- Activity Probability: a parameterised version of the activity value γ that can go from deterministic $\gamma \to \infty$ to probabilistic $\gamma \to 0$.

Each agent chooses the activity which maximises the activity probability according to the parameter γ. The parameter α accounts in a way the ”selfishness“ of the agent because when $\alpha = 1$ the agent will choose the action that will produce the most likely reaction from Alter whereas when $\alpha = 0$ the agent will choose the action that Alter will expect more.

The problem is then to measure Social Order and see if the dyadic condition is able to produce high values of social order. There are 2 different points of view: a system view and an individual view.

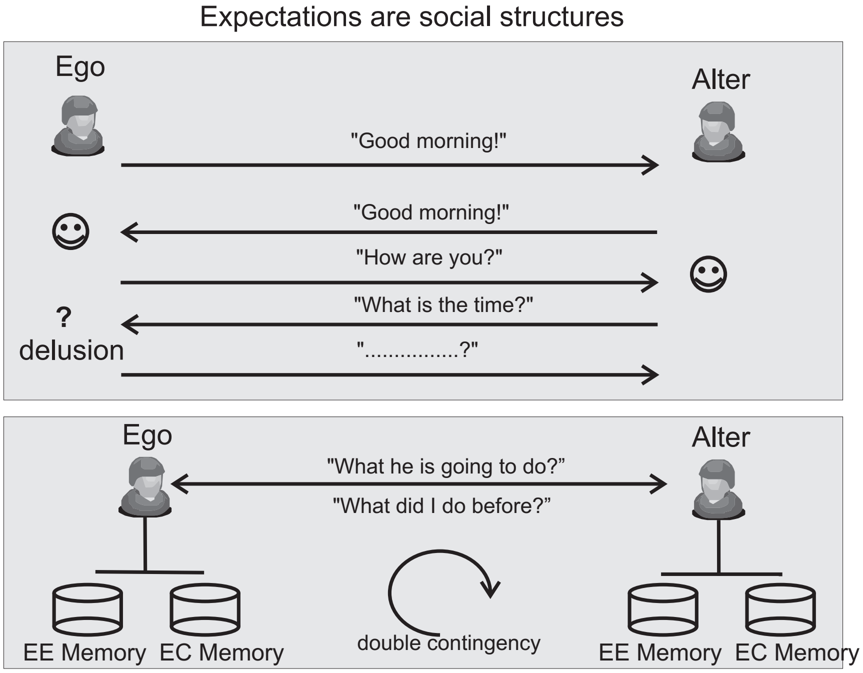


Figure 2.12: Upper block: an example of mutual expectation in a conversation. Lower block: the simulation model with the Expectation-Expectation memory and the Expectation-Certainty model.

At the individual level we can re-use the EC function to compute how certain an agent is when it selects a message. The average certainty O_{AV} has a high value when certainty is high and thus indicates high social order.

At the system level we can measure:

- the average number of different activities N_D selected during the time interval: the lower the number the higher the order. An observer will deduce that high social order is achieved if agents always select the same activities out of a vast selection set.

- predictability of an activity O_p or social integration: it measures how predictable an activity of a randomly drawn agent Ego is, given the activity presented on the sign by another randomly drawn agent Alter.

We can interpret the O_p value as an index of the pattern formation in the behaviour of action selection. The actors formed a closed system of interaction because they are required to develop mutually predictive trust, this closure indicates a first order separation degree between a system and its environment.

2.3.2 Results

For the dyadic case social order as measured either by N_D and O_{AV} emerges for any parameter setting in α, γ, N with stable activity patterns following robust to small disturbances. Measuring social integration for the dyadic case is trivial because there are only 2 agents who interacted with each other and thus it will be $O_p = 1$.

The big challenge is then to scale up the dyadic case to the multi agent case. If we have a population of few M agents and we choose a random selection strategy to pair interactions, social order is high in terms of O_P because is possible to predict each agent's reaction with a high degree of accuracy. However at the individual level O_{AV} is low because each agent is using the same memory to predict interactions with different agents. As a consequence increasing M decreases the O_P system order.

Dittritch discovered that there are 2 changes required to produce high social order with an increasing number of agents:

- agents must calculate Expectation-Expectation from observation of the interaction of other agents

- agents must use only Expectation-Expectation for activity selection ($\alpha = 0$)

At the individual level agents are cognitive entities able to perceive, memorise, generalise and to make predictions. For society to emerge they must be able to observe the interactions between others.

2.3.3 Discussion

The results of such a model were quite promising but there are also some weaknesses:

- the system operates exclusively on a symbolic communicative system

- the system does not distinguish between actions which manipulate the environment and communications

- the system is discrete and cannot operate in the analog domain

The mentioned issues were the main drives for the model described in this thesis because if such a social system needs to be implemented in the real world, the agents or robots needs to have a separate layer, one for the actions in the environment and one for the communicative events which are essentially symbolics.

Another desired property is the use of analogic based controllers which can then be implemented in very fast reactive electronic controllers or also digitized and implemented on a micro controller. In the next section, I am going to describe the framework used to achieve such targets.

2.4 Agent Based Modelling

This Section contains a summary of current modelling strategies for the simulation of software agents which interact with each other like animal groups or human societies.

2.4.1 SWARM and ABM

The simulations of artificial agents in this Thesis were done by using an ABM approach. In literature there is, unfortunately, a conflict in notations as well as several debates in the differences between MAS, ABM (agent based models) and SWARM intelligence. The expression SWARM intelligence (Beni and J.Wang, 1989) was introduced by Gerardo Beni and Jing Wang in 1989, in the context of cellular robotic systems. SWARM refers to a form of collective behaviour exhibited by groups of homogeneous animals: flocking is the swarm behaviour of birds, herding is the swarm behaviour in quadrupeds, schooling is the swarm behaviour in fish. The original term was introduced Dr.Marco Dorigo:

> Swarm intelligence is the discipline that deals with natural and artificial systems composed of many individuals that coordinate using decentralized control and self-organization. In particular, the discipline focuses on the collective behaviours that result from the local interactions of the individuals with each other and with their environment. Examples of systems studied by swarm intelligence are: colonies of ants and termites, schools of fish, flocks of birds and herds of land animals. Some human artefacts also fall into the domain of swarm intelligence, notably some multi-robot systems, and also certain computer programs that are written to tackle optimization and data analysis problems (Dorigo and Birattari, 2007).

Swarm intelligence has been used to model the clustering behaviour of ants, the nest building behaviour of wasps and termites, flocking and schooling in birds and fish. Particle swarm optimisation (Kennedy and Shi, 2001) is a population based stochastic optimisation technique for the solution of continuous optimisation problems and therefore is used more in the realm of functional mathematics.

The simulations used in this Thesis cannot be considered as Swarm simulations because the agents are active learners and share some features with human like societies rather than animal societies like ants. Therefore the author will use the term ABM for referring to the actual implementation. Nevertheless, there are some human behaviours like panic behaviour that can be described as herding behaviour and thus can be still classified as Swarm behaviour (Bonomi et al., 2009; Georgoudas et al., 2006; Kirchner and Schadschneider, 2002; Varas et al., 2007; Zong et al., 2010). The choice of the ABM term for my simulations in this Thesis was made for clarity and to classify the work in this fast growing sector.

Research in ABM involves the investigation of autonomous, rational and flexible behaviour for entities such as software programs or robots, and their interaction and coordination in several areas including: robotics (Hiroaki Kitano, 1997), information retrieval and management (Wooldrige., 2002; Guttman et al., 1999) and simulation (Gilbert and Conte, 1995). When designing agent systems, it is impossible to anticipate all the potential situations an agent may encounter and to optimally specify an agent behaviour in advance. Agents therefore have to learn from, and adapt to, their environment, especially in a multi-agent setting. This is especially true for multi-agent systems where in many cases global behaviour emerges rather than being pre-defined.

ABM can be seen as the natural extension of the Ising model (Ernst, 1925) or Cellular Automata-like models (Wolfram, 1994) which have been very successful in the past decades at describing various physical phenomena like Ferromagnetic materials.

One important characteristic of ABMs, which distinguishes them from Cellular Automata, is the potential asynchrony of the interactions among, and between, agents and their environments. In ABM agents typically do not simultaneously perform actions at constant time-steps, as in CAs or boolean networks. Rather, their actions follow discrete-event cues or a sequential schedule of interactions. The discrete-event setup allows for the cohabitation of agents with different environmental experiences. Also ABMs are not necessarily grid-based like the Conway's game of life (Gardner, 1970) and can also simulate analogic agent controllers. In particular, the richness of detail one can take into account in ABM makes this methodology very appealing for the simulation of biological, social and economic systems; where the behaviour and the heterogeneity of the interacting components are not safely reducible to some reduced models or differential equations.

In essence the two reasons for the choice of ABM in this thesis are:

- simulation in real time of parallel analog or discrete processes
- flexibility in the implementation of hardware based robots

To make a concrete case in Appendix 5.7, the software agent was implemented on a robotic kit called Lego and on a small robot called Pololu 3π and in Appendix 5.2 there are some code examples taken from the Enki simulator which is able to describe the mechanical and electronical properties of a real Alice or Kephera robot including for example the motor noise and the jerkiness of the stepper motor.

2.4.2 Single agent VS multi agent learning

There are two main approaches to ABM systems and learning:

- "single agent learning“: existing multi-agent machine learning algorithms are applied directly to single agents in a MAS setting. Consequently, multi-agent learning is only seen as an emergent property.
- ”multi agent learning“: agents need to cooperate and communicate in order to learn effectively.

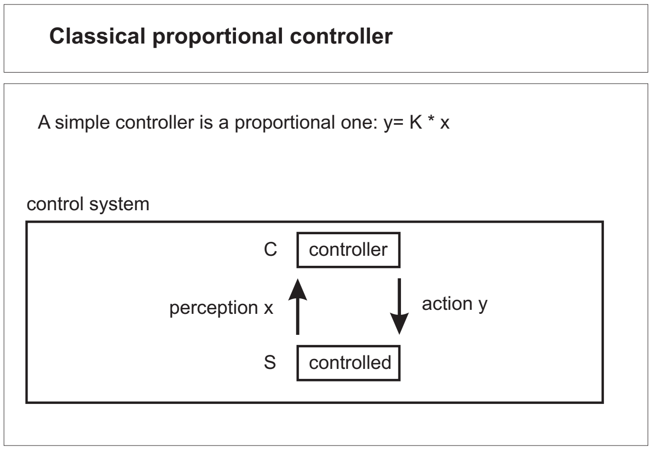
Single agent learning (Stone and Veloso, 1998; Porr and Wörgötter, 2006; Porr and Wörgötter, 2003) focuses on how one agent improves its individual skills, regardless of the domain in which it is situated. As discussed before, thanks to the closed loop property of such controllers the organism can still learn from the others by means of their motor actions which feedback into each others' inputs: each agent perceives the others as part of the environment feedback loop. Previous research studies have shown how is possible to create coordinated group behaviour with pure single-agent learning (Sugawara and Lesser, 1998). This is also the case, as we are going to see, for the model that is used in my research which contains also a communication layer through which agents learn to cooperate in a foraging task. Thus in my model each agent learns with a predictive controller (Porr and Wörgötter, 2006). We investigate the influence of communication in the Section 3.1 where the property of double contingency is used succesfully to reproduce social order in artificial societies.

2.5 Information theory for closed loop controllers

This section contains the essential background for the application of information theory to closed loop controllers. It will be used extensively for the computation of information measures in the following sections.

2.5.1 Introduction: closed loop controllers

In control theory there are two main approaches: feed-forward and feedback control. Feed-forward control is possible only when the law or transfer of the process I want to control is known. Natural systems are very hard to control using the open loop approach because they are dynamic non-linear processes and their parameters are subject to noise. Feedback control does not require the full knowledge of the process and is based on the principle of action and reaction. Adaptive controllers change their parameters according to the observed parameters of the process to control: most biological organisms use the control loop approach. The daily process of maintaining our body temperature is an example of such a system (Werner and Buse, 1989). Until today cybernetics has provided a lot of adaptive controllers for the closed loop: PID controllers (see Figure 2.13) and the Kalmann filter to mention the most important.



Classical proportional controller

A simple controller is a proportional one: y= K * x

Figure 2.13: A classical proportional controller described only by the gain K.

Artificial intelligence (AI) branched from cybernetics and used what was available from the previous approaches, but it realised that designing artificial agents for the real world using the previous approaches was not possible: agents should use biological inspired behaviour to operate in the real environment without any harm for other persons or entities. Hebbian learning, neural networks, Q-learning, fuzzy logic and many others are nowadays used in many artificial agent systems. However using these adaptive controllers raises a big concern: how can I assess the performance of the agents? The question is not as trivial as it seems: the agent adapts to the environment, thus in general a more complex environment produce a more complex behaviour (Nolfi, 2005). But agents essentially perform input control to keep their desired state. The only solution to that problem is to develop input based measures and not output based measures as I will demonstrate in Section 3.2. Input contains the output of the agent through the environment, therefore if the agent is learning, it means it is changing the inputs to achieve a desired state (for example avoiding a painful signal). This approach was introduced by Gibson and J. (1955) who rejected cognitivism and behaviourism for a more direct realism. He argued that organisms perform only input control, one of his most radical approach was the concept of "affordances" whereby the objects of our environment tell implicitly how they want to be operated. Although there are some cases where organisms perform open loop forward control, the general principles described by Gibson are valid. the general model is described in Figure 2.14.

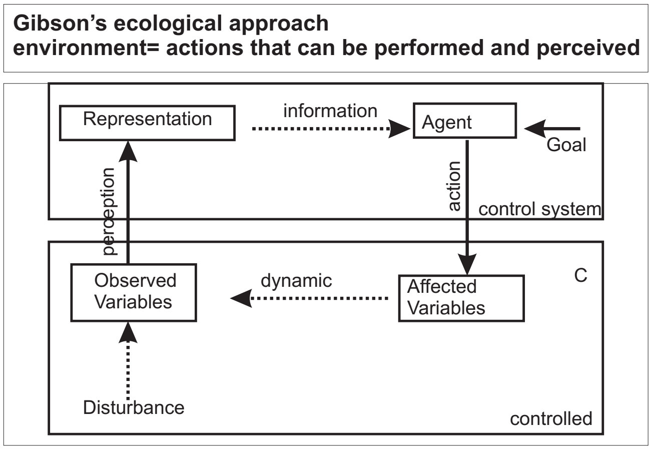


Figure 2.14: Model of perception as described by Gibson in 1955: the agent performs input control.

The theory of Shannon's information cannot be applied directly in closed loop systems because the information flow in an organism is asymmetric (see Figure 2.15). In the agent perspective the action is sensed via a representation of the controlled process.

In fact the traditional notion of Shannon entropy was extended by Touchette and Lloyd (2004) to closed loop systems by considering the sensory-motor loop as a communication channel which extends in time to propagate information from and to the environment. For instance Touchette and Lloyd (2004) revisited the idea of a controller as an actuation channel that transforms input states to desired output states. He also gives necessary and sufficient conditions for a system to be perfectly controllable and perfectly observable in terms of information and entropy. His model is a Bayesian network composed by a sensor channel S, an actuator channel A, the initial state of the system X and the final state of the system X'. The model as described in Fig. 2.16 both represented in Shannon's terms by their probability distribution matrix $p(a|s)$, where s is the input random variable of the channel and a the output random variable of the channel. The controller is modelled as a random variable C that takes an initial state X of the system to a final target state X'. A closed loop controller chooses a regulation $a_i \in A$ based on the state of the system $x_i \in X$, whereas an open loop controller chooses a regulation $a_i \in A$ that is independent of the sate of the system $x_i \in X$. Open loop control is thus different from a closed loop control in terms of mutual information $I(X, C)$ as described

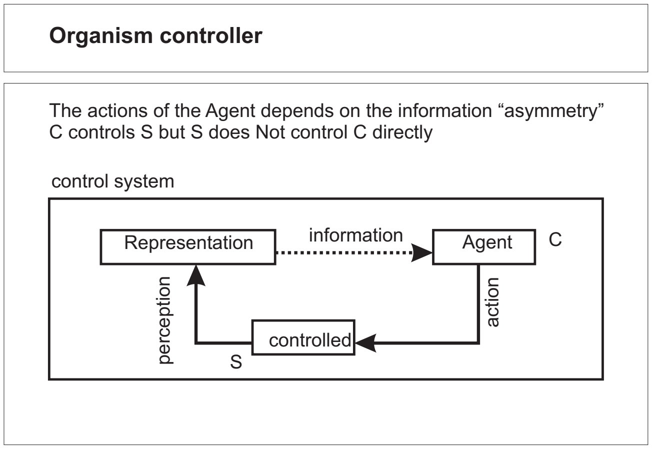

Organism controller
The actions of the Agent depends on the information “asymmetry”
C controls S but S does Not control C directly
control system
Representation
information
Agent
C
perception
action
controlled
S

Figure 2.15: The problem of information asymmetry in the closed loop control case.

in Eq. 2.2:

$$\textit{open loop} \quad : \quad I(X;C) = 0 \tag{2.1}$$

$$\textit{closed loop} \quad : \quad I(X;C) > 0 \tag{2.2}$$

The controller can be extended for the predictive case introduced in section 2.2.2, by adding an additional variable Y which is the predictor of X as in Fig. 2.17: Y conveys information about X and is used by the controller to infer the temporal relation between the predictor Y and the reflex X. This Bayesian model is investigated more in depth in section 4.3.7.

Other important tools were provided by Tishby et al. (1999) with the the information bottleneck method which considers the ability of learning in predictive controllers related to the compression achieved when two or more signals are mutually dependent.

The first practical application of information measures to closed loop controllers was introduced by Polani et al. (2004) who used the Tishby's framework to implement an information based controller as described in Fig. 2.18. The approach used is similar to the one described in Fig. 2.15 but with an optimization value provided by the information flow: the controller's transfer function is a model of the environment and the goal of the controller consists in maximizing the information flow from the motor output to the sensory input. With this very general approach the controller is able to achieve meaningful states without even defining a task objective. The disadvantage of the approach is that the controller needs to have a good model of the environment and needs to be able to do simulations and choose the best path to maximize the information flow.

A more general approach was introduced by Pfeifer et al. (2008) who used several information based metrics to follow the learning curve of the robot in a visual foveation task: the robot reduces the entropy around the centre of focus. Pfeifer et al. (2008) describe a closed loop system as a system that decreases the entropy, increases the information structure (statistical regularity), decreases the complexity.

A similar approach was used by Ay et al. (2008) to modify the proportional coefficient

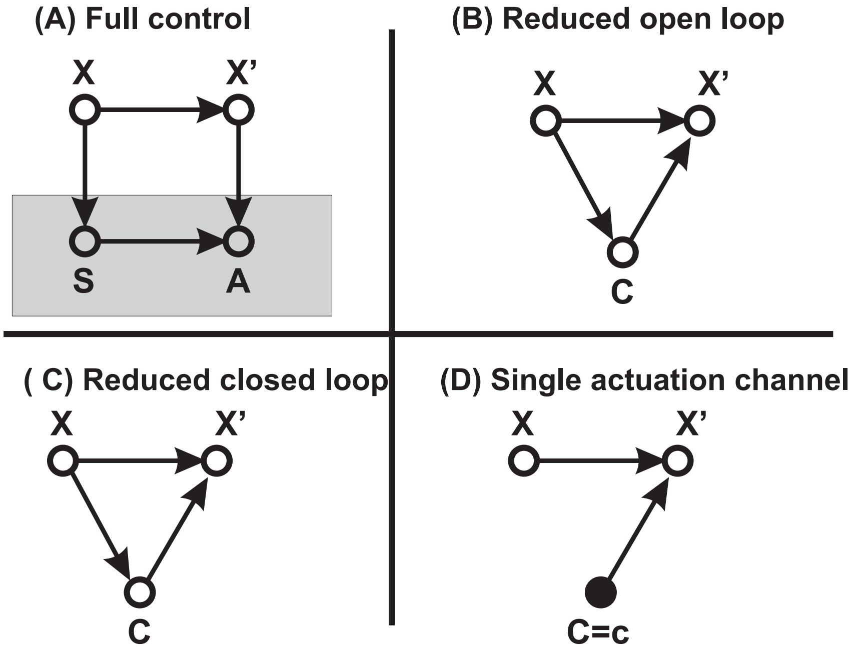


Figure 2.16: Bayesian formulation of open and closed loop. A) Full control model. B) Reduced open loop model. C) Reduced closed loop model. D) Single actuation channel model. The only way to distinguish open loop from closed loop is by means of Eq. 2.2.

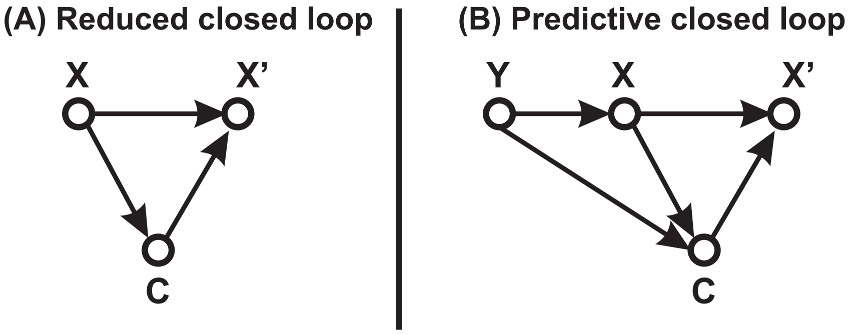


Figure 2.17: Extended Bayesian formulation. **A)** open loop case with sensor collapsed into controller $S = C$. **B)** closed predictive loop with Y predictive signal on X.

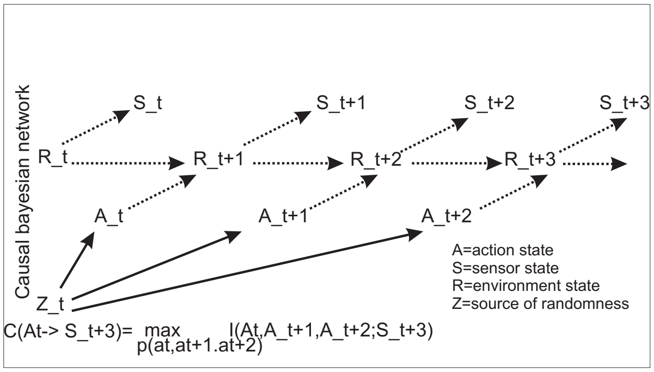


Figure 2.18: The empowerment is defined as the maximum information flow from the action A_t to the sensors S_{t+1} via the environment R_t. The source of uncorrelated randomness Z_t is useful to assess the controllability when the organism is removed from the control law.

in the controller's function to allow an optimal exploration strategy of an environment with obstacles: the mutual information between past and future is used to tune the gain c so that the maximum of the predictive information defines the best exploratory behaviour.

The next section contains an overview of Ashby's framework which will be used in this thesis.

2.5.2 Regulation and entropy

The main contribution earlier than Tishby to the cybernetic theory of controllers was produced by Ashby in 1956 and it will be used heavily in this thesis. Ashby (1956) defined clearly that the essential feature of a good regulator is to block the flow of variety from disturbances to essential variables. Ashby uses variety precisely as the number of different states a variable can be and so it relates to Shannon's entropy: if a variable has a variety of 4 states, then it can be described with 2 bits in terms of Shannon's entropy. Thus variety and entropy can be used alternatively. I use the same notation by Ashby so that it will be more clear how those concepts can be applied successfully to predictive learning:

- D is the domain of disturbances from the environment like a threat for an organism.

- E is the domain of the essential variable, can be partitioned in $E = \eta \cup \overline{\eta}$, where η is a partition of desired states or goals of the organism and its complementary partition $\overline{\eta}$ represents the non-desired states.

- R is the domain of available regulations that the organism can perform

- T is the domain of the possible states of the environment.

- F is the combination of R and T.

The disturbance D tends to drive E outside the set of desired states η. For open loop control systems the relationship between the mentioned variables is shown in Fig.2.19(A). Fig.2.19(B) describes how the disturbance is absorbed by the regulator and the environment to keep a desired state.

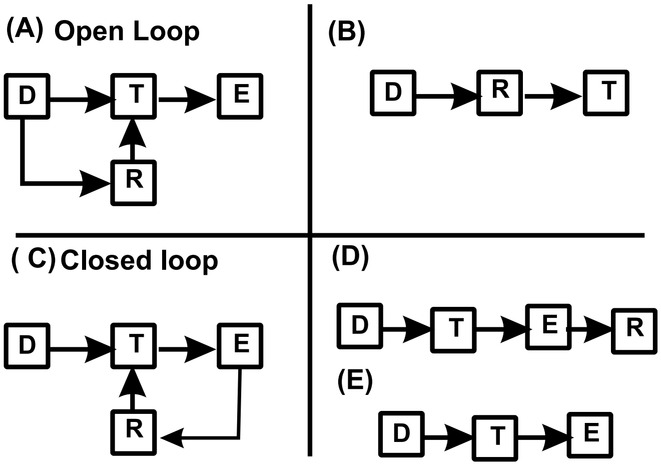


Figure 2.19: Ashby's law of requisite variety (Ashby, 1956) applied to the open loop case (A),(B) and the closed loop case (C),(D).

Figure 2.19(C)(D) describes how the disturbance is absorbed by the regulator and the environment to keep a desired state in a closed loop configuration.

The problem of regulation is defined as follows: given E,η, T, and D, to form the mechanism R so that R and T, coupled, act to keep E within η.

2.5.3 Direct regulation

Ashby described direct regulation as a control strategy similar to what engineers call open loop control. The update rule for the direct regulation of Figure 2.19(A) follows:

- D generates a disturbance $d(t)$
- $d(t)$ is the input to R which outputs $r(t)$
- the 2 values $d(t), r(t)$ are inputs to T that produces $e(t)$
- the value $e(t)$ is a state in E which can be a desired or non desired state

In the animal world, regulations in simple animals are direct: the organism reacts to the disturbance D before it affects E. A good example is in the life cycle of frogs: tadpoles reacts to a touch stimulus (for example when a person poke them with a finger) with a swimming reaction opposite to the direction of the stimulus. After some time the tadpole will stop swimming: there is no way for the animal to check if the disturbance was still there.

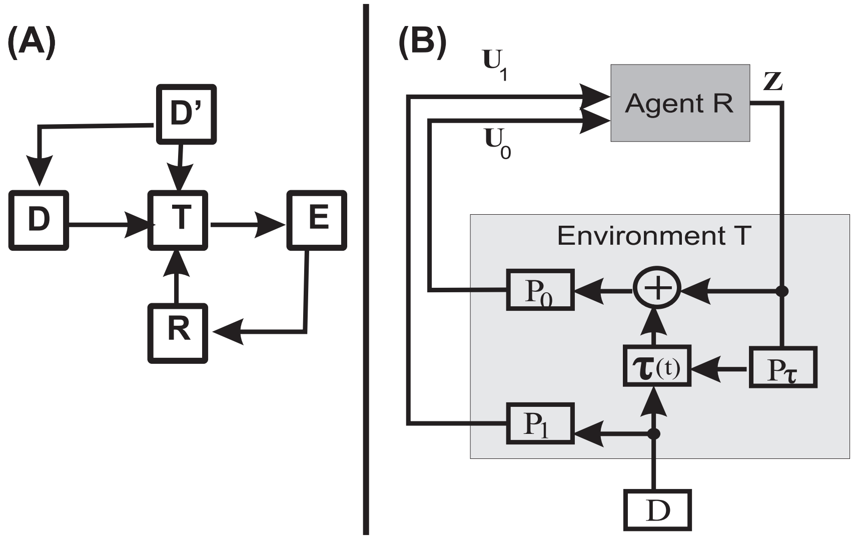


Figure 2.20: Ashby law of requisite variety applied to the ICO learning controller

Most often in the animal world, the R's action cannot be completed before the output of T is known: the regulator does not know the disturbance directly but only through the environment or after the organism has experienced the disturbance.

2.5.4 Closed loop regulation

The closed loop regulation is defined when R receives its input from E and not from T as in Figure 2.19(C). The regulator does not receive directly the disturbance but only after it passed the environment or the organism.

The special case of a predictive controller was not considered by Ashby and therefore it will be necessary to formulate the necessary conditions for learning. A possible extension of the Ashby's variety for the ICO controller is in Fig. 2.20 on the right side there is the ICO controller of the robot and on the left side there is the corresponding machine state model. In Fig. 2.20(A) the disturbance D' is a predictor of D and goes through the environment T affecting the essential variables E of the controller. In Fig. 2.20(B) the diagram of the ICO controller This novel approach to predictive controllers is going to be investigated in Section 3.5. In the rest of this section, the original formulation of Requisite Variety is introduced.

2.5.5 The law of requisite variety

Please note that with the same notation D,E,R,T,F in the diagrams, I refer to the determinate or indeterminate (Markovian) closed transformation whose allowed states are present in the corresponding domains. **Definition of regulation:** an organism is a perfect regulator if is able to keep the essential variables in a desired set η in spite of the disturbances. **Regulation**

blocks the flow of entropy: if F is a regulator, the insertion of F between D and E decreases the variety that is transmitted from D to E. How do we measure the performance of R as a regulator? **The function of the regulator R is to reduce the entropy that is transmitted from D to E allowing the organism to be in the η partition of desired states.** Thus if F is missing the organism will likely experience the entire set E, with a variety $H(E)$ but when F is introduced, the variety will be reduced to $H(\eta) < H(E)$. Thus a perfect regulator will prevent the organism from knowing in what state the disturbance was: the information channel that goes from the disturbance to the essential variables is blocked totally by the regulator.

The law of requisite variety in terms of Shannon's entropy:

> The law of requisite variety says that R's capacity as a regulator cannot exceed R's capacity as a channel of communication. It can be formulated in Shannon's terms assuming that: D is the noise that is being transmitted to E the essential variables of the organism by means of T the environment, R the regulator is a correction channel whose input is D and whose output is T whose role is to reduce the variety in the E channel. In an ideal case $H(E) = 0$ such that $H(D) = H(R)$, the regulator must have the same variety as the disturbance.

2.5.6 First law of requisite variety

Let D, R and E be three random variables. **Hypothesis:** when R is given, the entropy of E cannot be less then that of D: $H(E|R) \geq H(D|R)$. Then the role of the organism is to achieve maximum control over its internal variable E, thus reducing the uncertainty. The minimum possible regulation that can be achieved is $H(D) + H(R|D) - H(R)$:

$$H(E) \geq H(D) + H(R|D) - H(R) \tag{2.3}$$

Corollary 1: if R is a determinate function of D: $H(R|D) = 0$, the minimum entropy of E is $H(D) - H(R)$. The first law of requisite variety says that E's entropy can only be reduced by an equal increase in R's variety. Proof:
For the chain rule of entropy:

$$H(R, D) = H(D) + H(R|D) = H(R) + H(R|D) \tag{2.4}$$

substitute $H(E|R)$ for $H(D|R)$ in previous equation gives:

$$H(D) + H(R|D) \leq H(R) + H(R|E) \tag{2.5}$$

$$H(D) + H(R|D) \leq H(R, E) \tag{2.6}$$

$$H(R, E) \leq H(R) + H(E) \tag{2.7}$$

$$H(D) + H(R|D) \leq H(R) + H(E) \tag{2.8}$$

$$H(E) \geq H(D) + H(R|D) - H(R) \tag{2.9}$$

The corollary was demonstrated in Ashby (1956).

2.5.7 Second law of requisite variety

Let D,R and E be three random variables. Hypothesis: when R is given, the entropy of E cannot be less then that of D minus a constant K: $H(E|R) \geq H(D|R) - K$. Then the minimum entropy of E is $H(D) + H(R|D) - H(R) + K$:

$$H(E) \geq H(D) + H(R|D) - H(R) + K \tag{2.10}$$

The constant K is here used to model the “handicap” of the organism.

Chapter 3

Research work

3.1 Introduction: Social Modelling of Artificial Agents

The aim of this section is to develop an Agent Based Model which satisfies the autopoietic property of Social Systems as introduced in the previous section 2.1. The assumption here is that in an Artificial Society subsystems are formed to reduce uncertainty. Uncertainty is faced by agents when learning in their environment. The simplest learning algorithm is an appropriate reflex which guides the agent from or to a certain object, for example a wall or food, as described in the previous section 2.2.3. Learning enables the agent to anticipate reflexes and to generate anticipatory behaviour as discussed in the previous section 2.2.2. This, however, poses a problem because when all agents learn, they change their behaviour all the time which renders them more and more unpredictable to each other (Luhmann, 1995). Luhmann (1995) proposed that the creation of subsystems will overcome this problem. Within these subsystems, agents perform more predictably, by reducing their behavioural complexity. These subsystems are formed by adaptive communication between the agents which seems to be essential to form such subsystems. Both the behaviour and communication is learned by the agent and is not imposed on the agent. The goal or motivation of each agent is to collect food, keep it and eat it until consumed. Every agent broadcasts its hunger state, which can be used by other agents, into the world. This results in two subsystems where agents in the first collect food and in the latter steal food from others. The section is structured in this way: description of the agents, world and signals involved, the learning rule used, agents' behaviours, sub-system formation and effects of different communication strategies followed by a conclusion.

3.1.1 Methods: A Model of the World

The simulation model is composed of a 2 dimensional world bounded by walls. It contains two different objects: agents and food sources. The agents, referenced by their position as $a_j(t)$,where a_j has 2 components (x,y coordinates indexed by $a_{j,x}$ and $a_{j,y}$), with $j = 1, .., N$. Agents move with a differential drive system named after Braitenberg (Braitenberg, 1984). Food sources are disks located at fixed position $f_j(t)$ with $j = 1, ..., M$. They can produce constant food or limited food.

Agents have different sensors which enable them to sense obstacles, other agents' presence

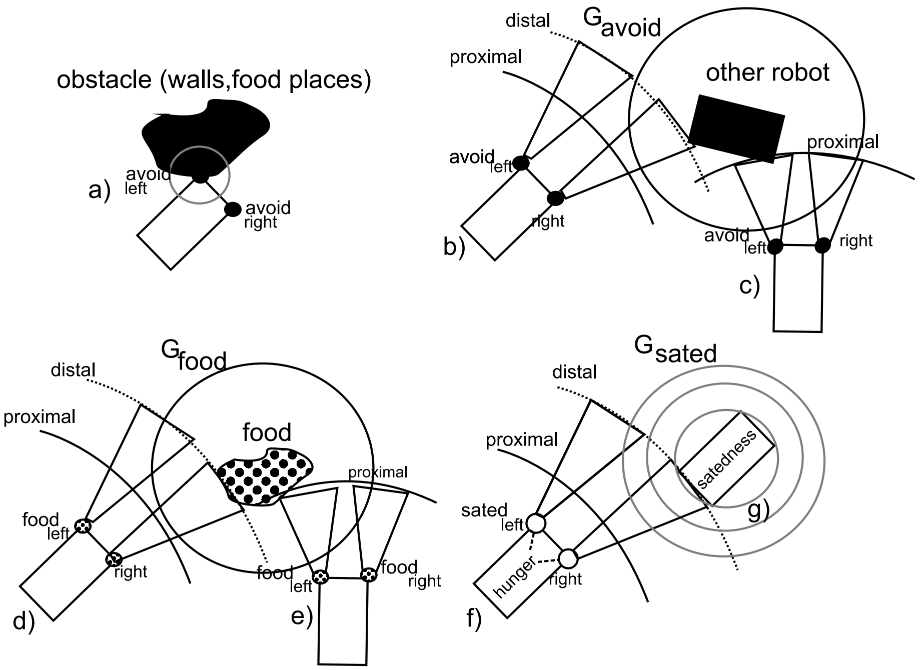


Figure 3.1: Overview of the different signals used in this simulation. Circles labelled with G (avoid,food,sated) represent uniform potential fields, circles on the robot's front are input sensor, cones irradiating from them represent the field of view of sensors, proximal and distal lines represent sensors range. Case a): an agent touches a food source or a wall with its proximal left avoidance sensor $avoid_{left,prox}$, a proximal signal is generated. Case b): an agent reads the potential field G_{avoid} produced by another, with its right distal sensor $avoid_{right,dist}$. Case c): an agent reads the potential field G_{avoid} produced by another agent, with both left and right proximal sensors $avoid_{left,prox}, avoid_{right,prox}$. Case d): an agent reads the potential field G_{food} produced by a food source, with its right and left distal sensors $food_{right,dist}, food_{left,dist}$. Case e): an agent reads the potential field G_{food} produced by a close food source, with both left and right proximal sensors $food_{left,prox}, food_{right,prox}$. Case f,g): a hungry agent f (with $Hunger = 1$) reads the satedness signal G_{sated} produced by the sated agent g.

and others' broadcasted state of satedness, at different ranges (proximal and distal see Fig. 3.1). Every object labelled with a certain index j produces a signal carried by a uniform potential field $G_{j,type}$ with a limited range, which is sensed by the corresponding sensor type (type can be avoid,food or sated). The potential field $G_{j,type}$ is described by the equation of a circle which is centered on x_0, y_0 with a r radius:

$$G_{j,type} : (x - x_0)^2 + (y - y_0)^2 = r^2 \tag{3.1}$$

Every geometric point x, y including a sensor or object which falls inside the circle $G_{j,type}$ assumes a unitary value. The signals from the proximal sensors (x_0) are originally used to drive the agents reflexes which can either be avoidance or attraction. The signals from the distal sensors are used for learning so that the agent is able to generate anticipatory reactions instead of the reflexes as introduced in section 2.2.2.

In the next section I am going to describe the learning algorithm enabling the agent to replace the reflexes with the predictive actions. Once the learning algorithm is described, I will describe the different reflexes and possible anticipatory reactions.

3.1.2 Methods: ICO learning module

The input correlation learning rule of Porr and Wörgötter (2006) is a Hebbian learning rule, it is unsupervised and performs a confounded correlation between a predefined reflex signal (x_0) and a reflex predicting signal (x_1). Hence, this learning algorithm identifies and exploits causalities between temporal sequential signals. The ICO learning algorithm was chosen because it is one of the simplest,fastest and computationally efficient approach to temporal learning. It can also be easily implemented in analogic and digital systems without any particular modifications.

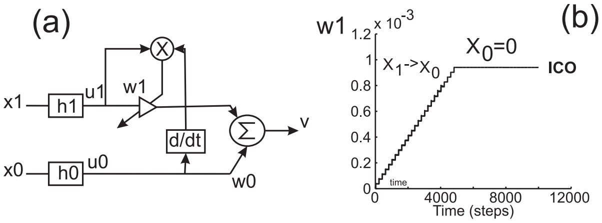


Figure 3.2: Figure (a) shows the ICO learning basic block composed by 2 inputs x_0, x_1 filtered by h_0, h_1 and the output v. The reflex is x_0 with a fixed weight and the predictor is x_1 with a variable weight. Figure (b) shows the weight change of w_1 during time. At the beginning $w_1 = 0$, then for 5000 simulation steps x_1 anticipates x_0 and the w_1 grows until 1.0. After 5000 simulation steps reflex is suppressed $x_0 = 0$ and w_1 stabilises to $1 \cdot 10^{-3}$.

Figure 3.2 shows the ICO learning block which has two inputs x_0,x_1 from the agent's sensor that are filtered by low pass filters h_0, h_1:

$$h(t) = \frac{1}{b}e^{at}sin(bt) \tag{3.2}$$

$$a = -\pi\frac{F}{Q} \tag{3.3}$$

$$b = \sqrt{(2\pi F)^2 - a^2} \tag{3.4}$$

F is the oscillation frequency and Q the quality factor. The low passed signals $u_i(t)$ are transferred with weight w_i to the output neurons (for more details see Appendix 5.1).

$$u_1 = h * x_1 \tag{3.5}$$

$$u_0 = h * x_0 \tag{3.6}$$

where $*$ is the convolution operation which implement the filtering operation. In the output neuron the output $v(t)$ is calculated by summing up all incoming signals according to their weights:

$$v(t) = w_0 \cdot u_0 + w_1 \cdot u_1 \tag{3.7}$$

which represent the input for the motor system. The unsupervised character of the ICO learning rule is reached by the synaptic weight w_1 to be adapted by the weight change rule:

$$\frac{\partial w_1}{\partial t} = \mu u_1 \frac{\partial u_0}{\partial t} \tag{3.8}$$

The weight change is dependent on the derivative of the reflex input signal u_0, the input signal u_1 and a learning rate μ. The learning rule has been shown to be useful for avoidance and attraction mechanisms and has fast and stable convergence (Porr and Wörgötter, 2007).

3.1.3 Methods: agent controller

For the sake of simplicity, the agent's neural controller is analysed block by block according to the requested behaviours (avoidance and attraction) in Figs. 3.3,3.5,3.6. The core is composed of two ICO neurons, labelled with L-eft and R-ight, connected to the motor outputs left and right. Both ICO neurons have a constant bias input B with weight 4.0 that makes the robot move forward if inputs are absent. Rectangular blocks labelled with L,R are low pass filters, with parameters F, Q referred to equations 3.3,3.4. Synaptic weights of the ICO block in Fig. 3.2(a) are labelled with W capital letter and two pedex that indicate the weight type (dynamical for the predictor and fixed for the reflex) and the synapse position (left,right). ICO neurons have recurrent synaptic connections, labelled as $W_{R2L}, W_{L2R}, W_{selfR}, W_{selfL}$ to implement a hysteresis effect, which causes the controller to not instantly follow signals (as in Hülse 2004; Hü and Pasemann 2002). It means reactions on an incoming signal are time shifted. This is useful to enable agents to escape from acute angles: if an agent incurs in an concave acute angle and has not a hysteresis mechanism, will generate a closed trajectory (loop), turning left and right alternatively.

3.1.4 Methods: avoidance behaviour

Agents and walls are obstacles. Agents produce obstacle signals (see Fig.3.1 (a) for obstacles and Fig.3.1 (b),(c) for other agents). Every agent $a_j(t)$ has a potential field associated (see Eq.3.1):

$$Gavoid_j(t) = G(x - a_{j,x}(t), y - a_{j,y}(t)). \tag{3.9}$$

that is sensed by the corresponding inputs of other agents $a_k(t)$ (with $k \neq j$) labelled as $avoid_{left,right}$. Walls and food sources do not produce G_{avoid}, so that agents sense them using proximal signals that are generated by collisions: when 2 distinct agents j and l collide at time t_0, such that $||a_j(t) - a_l(t)||_2 < D$ (D is the radius of the agent) an impulse is produced at $avoid_{l,r}$. The neural controller for the avoidance behaviour is shown in Fig. 3.3 (see also Stamm 2006), every ICO neuron (left and right) computes the following operations:

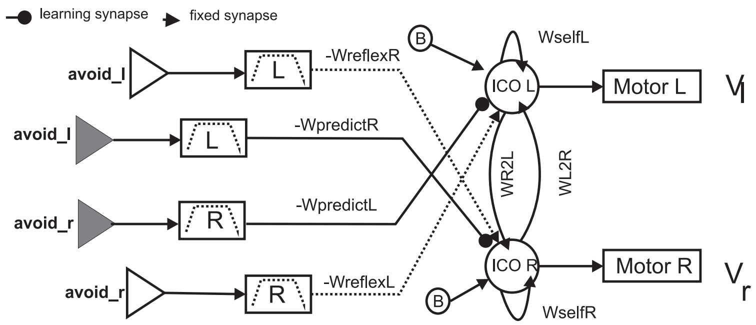


Figure 3.3: Avoidance network: ICO left and right are two neurons implementing the ICO learning rule, their output is a sigmoid and is connected (after normalization into $[v_{min}, v_{max}]$) to the motor speed commands. Grey triangles represents distal inputs, while white triangles represent proximal inputs. The learned synaptic weights are associated to the distal synapses (thick lines) while the fixed are associated to the proximal synapses (dotted lines). To produce a retraction behaviour left and right weights must be different such that $W_{predict,L} > W_{predict,R}$ and $W_{reflex,L} > W_{reflex,R}$, if $W_{predict,L} = W_{predict,R}$ and $W_{reflex,L} = W_{reflex,R}$ robot will just go back without turning.

$$\begin{aligned} ICO_L(t) &= B - h * avoid_r \cdot W_{reflexL} - h * avoid_{dist,r} \cdot W_{predictL} \\ &\quad + W_{selfL} \cdot ICO_L(t-1) + W_{L2R} \cdot ICO_R(t) \end{aligned} \tag{3.10}$$

$$\begin{aligned} ICO_R(t) &= B - h * avoid_l \cdot W_{reflexR} - h * avoid_{dist,l} \cdot W_{predictR} \\ &\quad + W_{selfR} \cdot ICO_R(t-1) + W_{R2} \cdot ICO_L(t) \end{aligned} \tag{3.11}$$

The parameters used for the weights, the bias and the recurrent connections are reported in the Appendix sections 5.2.4,5.2. The recurrent connections between the left and right ICO neuron, are necessary to implement a push-pull behaviour so that when the robot synchronously activates both the left and the right input, only one ICO neuron will dominate thus evoking a turn-back response. Connections between input synapses and ICO neurons (motor neurons) are negative to evoke a retraction. The motor output is calculated with a sigmoid activation function on the ICO neuron membrane as follow:

$$V_L = \frac{1}{1+e^{-ICO_L}} \tag{3.12}$$

$$V_R = \frac{1}{1+e^{-ICO_R}} \tag{3.13}$$

The weight update learning rule is calculated for the weights:

$$\frac{\partial W_{predict,L}}{\partial t} = \mu \cdot avoid_{dist,l} \frac{\partial avoid_l}{\partial t} \tag{3.14}$$

$$\frac{\partial W_{predict,R}}{\partial t} = \mu \cdot avoid_{dist,r} \frac{\partial avoid_r}{\partial t} \tag{3.15}$$

3.1.5 Methods: Agents and satedness communication

Every agent has an internal state: hunger and its complementary satedness.

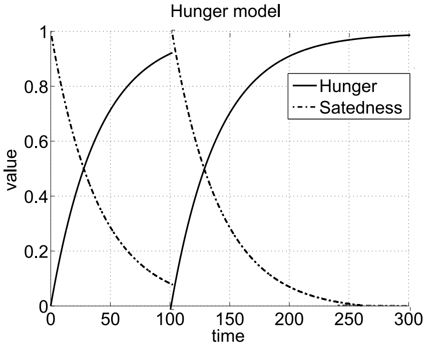


Figure 3.4: Hunger state in function of time. At time step 100 the agent touches a food place, thus its hunger state is reset to 0 and so its complementary satedness to 1.

The energy level of each agent is an exponential function of time:

$$H_{sated}(t) = \begin{cases} e^{-t/\tau_{starv}} & \text{if } t \geq t_b \\ 0 & \text{if } t < t_b \end{cases} \tag{3.16}$$

and complementary its hunger state is:

$$H_{hunger}(t) = 1 - H_{sated}(t) \tag{3.17}$$

where τ_{starv} is the starvation factor and t_b corresponds to the moment when an agent touches a food source:

$$|food_{l,prox,d_1}(t_b) + food_{r,prox,d_1}(t_b)| > \theta_F \tag{3.18}$$

or touches a sated agent (see Fig. 3.4):

$$|sated_{l,prox,d_1}(t_b) + sated_{r,prox,d_1}(t_b)| > \theta_A \tag{3.19}$$

and agent is touching an obstacle

$$|avoid_{prox,l}(t_b) + avoid_{prox,r}(t_b)| > \theta_O \tag{3.20}$$

where $\theta_F, \theta_A, \theta_O$ are thresholds for food, agents and obstacles respectively. In Fig. 3.5, internal state $H_{hunger}(t)$ is multiplied for $food(t)$ (left,right and proximal,distal) and $sated(t)$ (left, right and proximal, distal).

Satedness internal state of agent a_i, is broadcasted to other agents $a_k(t)$ (with $k \neq j$) by means of a potential field (see Fig.3.1(g) and Eq.3.1):

$$G_{i,sated}(t) = H_{sated}(t) \cdot G(x - a_{i,0}, y - a_{i,1}). \tag{3.21}$$

Agent $a_k(t)$ senses $G_{i,sated}$ (see Fig.3.1 (f)) with 2 reflexive inputs $sated_{l,prox,d_1}(t)$ and $sated_{r,prox,d_1}(t)$ whose difference feeds the reflexive input:

$$x_0(t) = sated_{l,prox,d_1}(t) - sated_{r,prox,d_1}(t). \tag{3.22}$$

and as predictive $sated_{l,prox,d_2}(t), sated_{r,prox,d_2}(t)$ whose difference feeds the predictive input:

$$x_1(t) = sated_{l,dist,d_2}(t) - sated_{r,dist,d_2}(t). \tag{3.23}$$

where $d_2 > d_1$. The equations 3.11,3.12 of the ICO neurons are added to the following synaptic inputs:

$$\begin{aligned} ICO_L &= -h * (x_0 \cdot H_{hunger}) \cdot W_{reflex,A,L} \qquad (3.24) \\ &\quad -h * (x_1 \cdot H_{hunger}) \cdot W_{predict,A,L} \\ ICO_R &= h * (x_0 \cdot H_{hunger}) \cdot W_{reflex,A,R} \qquad (3.25) \\ &\quad +h * (x_1 \cdot H_{hunger}) \cdot W_{predict,A,R} \end{aligned}$$

The reason for the sign inversion for the weights is that the inputs are differential and thus is necessary for the left ICO neuron have an input of opposite sign to the right ICO neuron. The weight update rule this time is:

$$\frac{\partial W_{predict,A,L}}{\partial t} = \mu \cdot x_1 \frac{\partial x_0}{\partial t} \tag{3.26}$$

$$\frac{\partial W_{predict,A,R}}{\partial t} = \mu \cdot x_1 \frac{\partial x_0}{\partial t} \tag{3.27}$$

Inputs for the attraction task are shown in Fig. 3.5. When for example: $x_0(t) > 0$ implies that a sated agent is on the left $sated_{l,prox,d_1}(t) > 0$, the neural controller produces $v_L < v_R$, agents turns left until $x_0(t) = 0$ that means either $x_0(t) = x_1(t)$ (a sated agent is in front) or $x_0(t) = x_1(t) = 0$ (no sated agent in front). A hungry agent, is producing $G_{i,sated}(t) = 0$ therefore other agents will be repelled since it is emitting only the G_{avoid} signal.

Aggressive agents If one wants to make an agent more aggressive $H_{sated}(t)$ can be multiplied for the distal sensors $avoid(t)$ left, right (in Fig. the H_{hunger} block can be introduced after each of the grey triangles).

$$H_{sated}(t) \cdot avoid(t) \tag{3.28}$$

It implies that when an agent is not sated, it will ignore the obstacle signal G_{avoid} produced by the other agent.

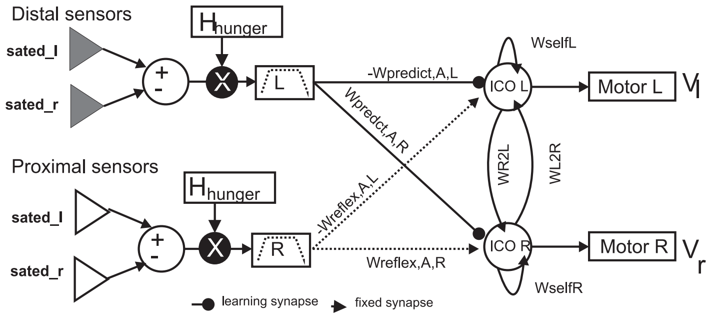


Figure 3.5: Attraction toward sated agents:two more inputs are added to the ico neurons. Grey triangles represents distal inputs, while white triangles represent proximal inputs. The learned synaptic weights are associated to the distal synapses (thick lines) while the fixed are associated to the proximal synapses (dotted lines). Synaptic weights must be equal in module and opposite in sign $|W_{predict,A,L}| = |W_{predict,A,R}|$ and $|W_{reflex,A,L}| = |W_{reflex,A,R}|$ where $A = agent$. Hunger internal state is multiplied for proximal and distal input difference

3.1.6 Methods: Food attraction

Every food source $f_j(t)$ with $j = 1, ..., M$ produces the signal (see Fig.3.1 (d),(e) and Eq.3.1):

$$G_{j,food} = G(x - f_{j,x}, y - f_{j,y}) \tag{3.29}$$

which is sensed by agents a_i by the inputs labelled as $food_{left,right}$ (see Fig.3.1 (d),(e)).

Constant food production Every food source contains a constant amount of food which means $G_{j,food}$ is always emitted.

Limited food production Every food source contains a limited amount of food modelled by the variable q that is decremented every time an agent touches the food place at t_b:

$$q_j(t_b + 1) = q_j(t_b) - \theta q \tag{3.30}$$

where (θq is fixed). When $q_j = 0$ the food source j is exhausted and food signal is suppressed $G_{j,food} = 0$. After a random period the food source is restored $q_j = 1$.

Required inputs for the attraction behaviour are introduced in Fig. 3.6 (see also Stamm 2006), for every ICO neuron an additional reflex is added:

$$x_0(t) = food_{l,prox,d_1}(t) - food_{r,prox,d_1}(t). \tag{3.31}$$

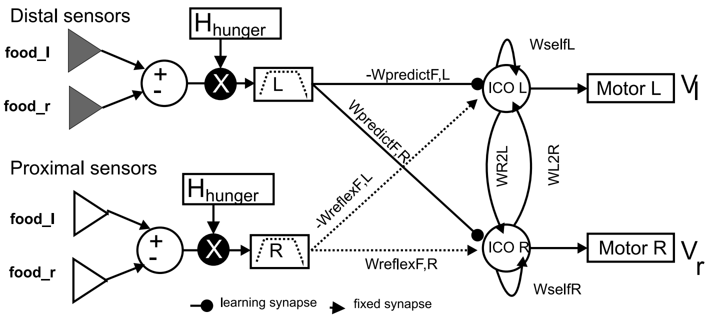


Figure 3.6: Attraction toward food: two more inputs are added to the previous network. Grey triangles represents distal inputs, while white triangles represent proximal inputs. The learned synaptic weights are associated to the distal synapses (thick lines) while the fixed are associated to the proximal synapses (dotted lines). Synaptic weights in the attraction task must be equal in module and opposite in sign: $W_{predict,F,L} = W_{predict,F,R}$ and $W_{reflex,F,L} = -W_{reflex,F,R}$ where $F = food$

x_0 it is the difference between the left and right proximal food input sensors. A predictive input is added:

$$x_1(t) = food_{l,dist,d_2} - food_{r,dist,d_2}(t). \tag{3.32}$$

$x_1(t)$ it is the difference between the left and right distal food input sensors. Thus $d_2 > d_1$ such that the distal food sensor values are predictive on the proximal food sensors. The equations 3.25,3.26 of the ICO neurons are added to the following synaptic inputs:

$$\begin{aligned} ICO_L &= -h * (x_0 \cdot H_{hunger}) \cdot W_{reflex,A,L} \qquad (3.33) \\ &\quad -h * (x_1 \cdot H_{hunger}) \cdot W_{predict,A,L} \\ ICO_R &= h * (x_0 \cdot H_{hunger}) \cdot W_{reflex,A,R} \qquad (3.34) \\ &\quad +h * (x_1 \cdot H_{hunger}) \cdot W_{predict,A,R} \end{aligned}$$

The reason for the sign inversion for the weights is that the inputs are differential and thus is necessary for the left ICO neuron have an input of opposite sign to the right ICO neuron. The weight update rule this time is:

$$\frac{\partial W_{predict,A,L}}{\partial t} = \mu \cdot x_1 \frac{\partial x_0}{\partial t} \tag{3.35}$$

$$\frac{\partial W_{predict,A,R}}{\partial t} = \mu \cdot x_1 \frac{\partial x_0}{\partial t} \tag{3.36}$$

When for example: $x_0(t) > 0$ implies that food source is on the left, the neural controller produces $v_L < v_R$, agent turns left until $x_0(t)$ becomes 0.

3.1.7 Controller summary

For clarity Fig. 3.7 contains the simplified but full structure of the controller whereby the two ICO neurons receive synaptic inputs from each synaptic input described before. There

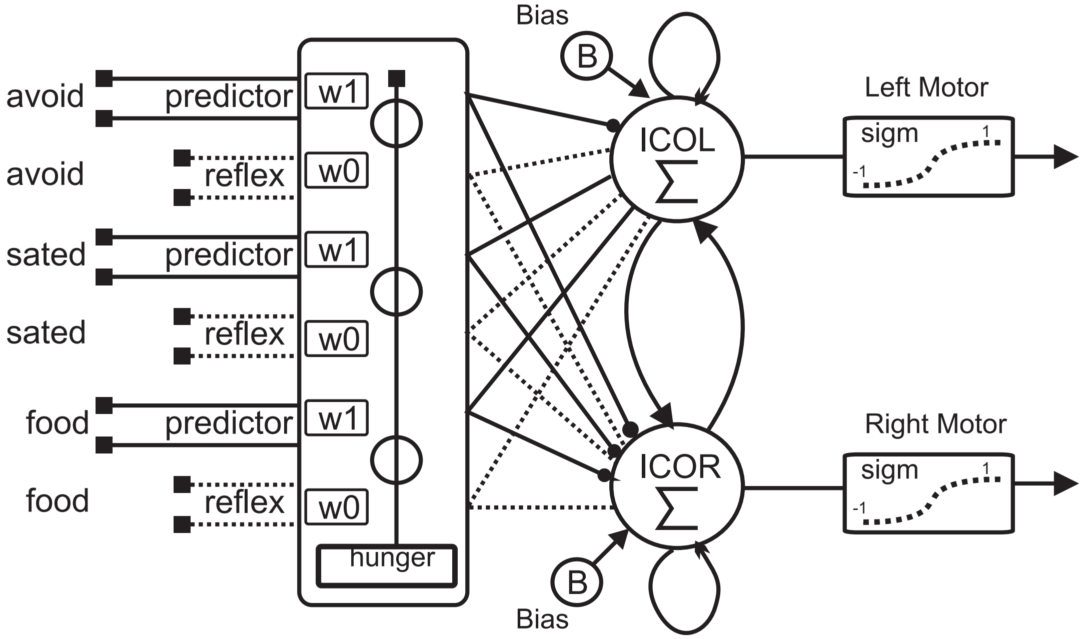


Figure 3.7: The robot controller implements the 3 behaviours by using a linear summation of all the synaptic inputs

are a total of twelve inputs because for every behaviour there are left and right sensors for the distal and proximal case. Because there are three behaviours, multiplied by 4 makes 12 parallel inputs. The inputs are then low pass filtered as described before and summed at the ICO neuron $\sum$. The left and right ICO neurons are also responsible for updating the weights for the predictors. The output of each ICO neuron is then fed into a sigmoid function which normalizes the output in the $[-1, 1]$ range for controlling the robot motors.

3.1.8 Broadcasting signal mechanism

The main feature of a social system is the production and use of signals. Thus each agent can emit the same field in equation 3.29 in the presence of a food source. A more detailed discussion about signalling strategies is in the Conclusion section 4.1.7. There are only two possible signalling strategies in my model, honest and dishonest strategies and are going to be described in the following sections.

Honest food proximal signalling

In this scenario agents signal the presence of food when they sense it using their proximal inputs and thus emitting G_{sated} when discovering a food source (as described in Eq.3.18). This "genuine" social behaviour might increase the foraging performance of the colony, but might cost the signaller because it can result in higher robot density and increased competition and interference nearby the food (i.e. spatial constraints around the food disk: only 9 agents can forage at same time). Thus, although beneficial to other colony members, signalling of a food location can constitute a costly act (Smith and Harper, 2003) because it decreases the food intake of signalling robots. By observing the trajectories, one can notice that agents tend to form lines around the food zones, this behaviour could boost the foraging performance as I will show later.

Dishonest food proximal signalling

In this case the agent "cheats" producing non predictable (using a uniform probability of emitting $p(e) = 0.6$) food signals when they are far away from the food zones ($food_{l,dist,d_2} = 0$ AND $food_{r,dist,d_2}(t) = 0$) and of course when they are not sated ($H_{sated}(t) < 0.2$ a threshold). The percentage of cheaters used in the test was: $10\%, 50\%$,and 80%. Doing so an agent reduces the competition around the food zones.

3.1.9 Results: analysis of formation in different cases

The following sections contain the most important test cases for our model:

- general overview about the sub-system property and the food signalling strategies
- a comparison of the food performance between adaptive communication and non communicative strategy
- an insight to the honest behaviour with unlimited resources
- an insight to the honest behaviour with limited resources and environmental changes

Sub-system formation and adaptive communication

During time agents learn to: avoid obstacles, search for food and search for other sated agents. Because all the inputs (see Fig. 3.7) are used in parallel and summed linearly for the motor behaviour, the weights will develop independently from each other in a competitive fashion. For example when an agent sees a closer agent with food and a food source, the outcome of the motor behaviour will depend on the weight status for each behaviour. Thus agents can be classified in 2 classes, seekers and parasites, according to their weights [1]:

$$\delta_{w,agent} = \frac{|W_{predict,A}(0) - W_{predict,A}(T_{sim})|}{W_{predict,A}(0)}. \tag{3.37}$$

$$\delta_{w,food} = \frac{|W_{predict,F}(0) - W_{predict,F}(T_{sim})|}{W_{predict,F}(0)}. \tag{3.38}$$

where the fraction is used to normalize the weight development, so in summary:

- **An agent is a seeker** $\delta_{w,agent} > \delta_{w,food}$, if it is more attracted by food places than other sated agents.

[1] $W_{predict,A}$ is the average of $|W_{predict,A,L}|, |W_{predict,A,R}|$ and $W_{predict,F}$ is the average of $|W_{predict,F,L}|, |W_{predict,F,R}|$

- **An agent is a parasite** $\delta_{w,agent} \leq \delta_{w,food}$, if it is more attracted by sated agents than food places.

The behaviour is only described by the predictive weights because the reflex weights are constant and thus do not provide a useful classification. This classification will then be compared to a subjective comparative analysis in section 3.5.7 where the Predictive Performance (shortened as PP) will allow the sub system analysis without the need of comparing the weights on each agent.

Sub-system formation is analysed for 2 important test cases:

- adaptive communication: the agents learn using both proximal and distal signals
- non adaptive communication: the agents do no learn from the G_{sated} food signal.
- silence: the agents do no produce the G_{sated} food signal necessary for the other agents to know whether food is available or not.

The silence condition does not imply that in Eq. 3.23 distal signals are suppressed but rather that agents will learn only when the neighbouring agent is enough close to be sensed by the far sensors. This implicate that learning will be happening still but only when robots are close to each other rather than via the broadcast field G_{sated}. Whereas in the non-adaptive communication the learning on the receiver agent is totally disabled and thus the weights for the parasitic behaviour are locked. For our simulation, a population of $N = 20$ agents is provided with $M = 4, 10, 18$ food sources sequentially. Population dynamic is observed for a total duration of $T_{sim} = 80000$ (time step $\triangle T = 0.01s$) and for a set of 100 simulations with different initial starting condition. The simulation time is long enough for the system to stabilise after an initial transitory phase. Each simulation is randomized regarding the food disk and agents positions, this will guarantee 100 different trajectories. Fig.3.8 resumes a total of 6 test cases: left column considers scarce resources ($M = 4$ food sources against $N = 20$ agents), right column considers abundant resources ($M = 18$ food sources against $N = 20$ agents). Each one of them reports the number of seekers (thick line) and parasites (dotted line) in function of time. The number of seekers $n_s(t)$ is complementary to the number of parasites $n_p(t)$:

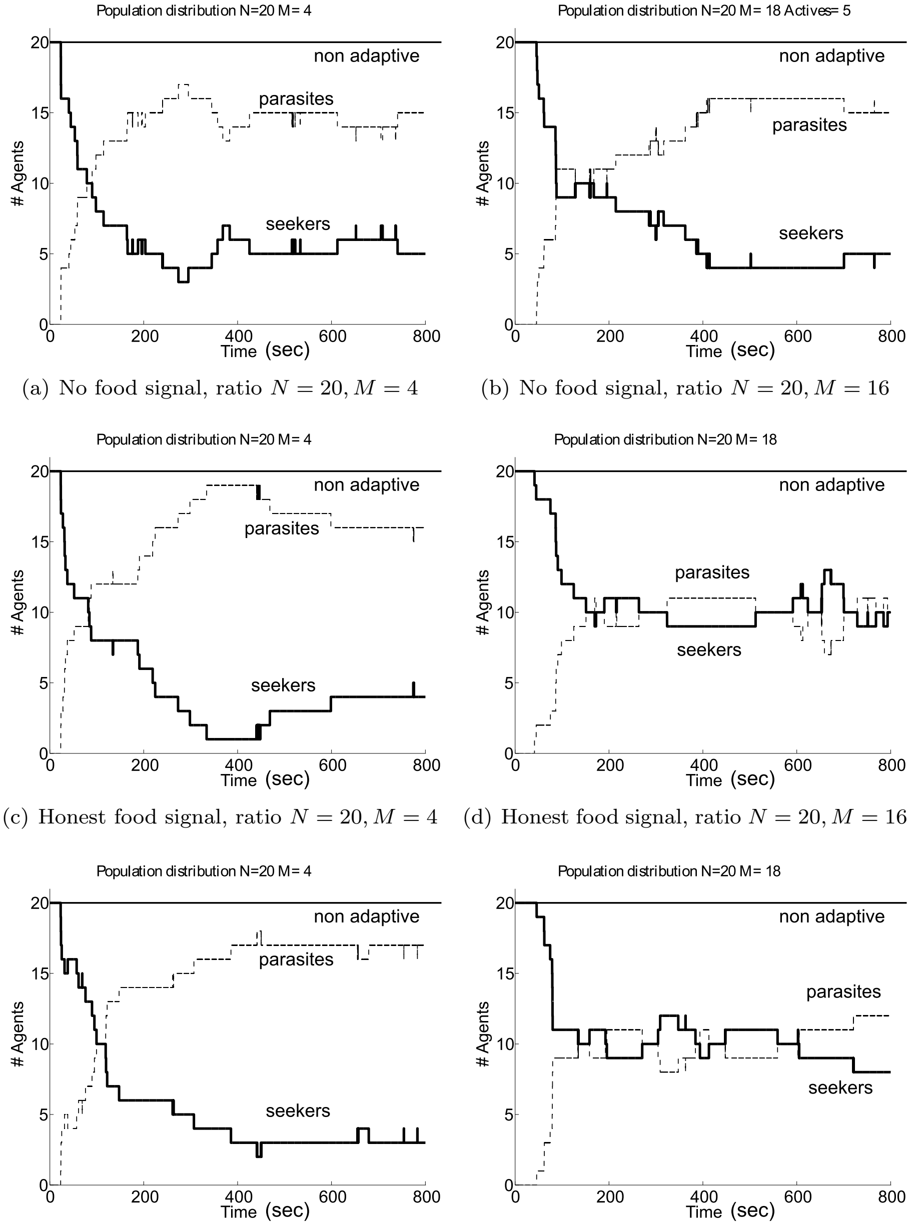
$$n_s(t) + n_p(t) = N \tag{3.39}$$

When the simulation is over $t = T_{sim}$ the ratio is still:

$$n_s(T_{sim}) + n_p(T_{sim}) = N \tag{3.40}$$

Considering the different cases in Fig.3.8, it can be said that :

- in non-adaptive communication: parasites are absent (also with aggressive configuration see condition 3.28), the explanation is trivial because the agents cannot learn to differentiate in the absence of a signal, therefore in all cases (a) to (f): $n_s(t) = 20$ and $n_p(t) = 0$ at any time.
- in adaptive communication a stable condition is achieved: the numbers of seekers and thus parasites stabilise after a transitory phase. For example in case (a) population distribution reaches an equilibrium: $n_s(t) = 5 \pm 1$ and $n_p(t) = 15 \pm 1$ with $t > 400s$. After 800 seconds (data not shown) the population distribution oscillates around the equilibrium.



(a) No food signal, ratio $N = 20, M = 4$

(b) No food signal, ratio $N = 20, M = 16$

(c) Honest food signal, ratio $N = 20, M = 4$

(d) Honest food signal, ratio $N = 20, M = 16$

(e) Dishonest food signal, ratio $N = 20, M = 4$, 18 cheaters

(f) Dishonest food signal, ratio $N = 20, M = 16$, 18 cheaters

Figure 3.8: A typical single run of the system which shows different cases. Population distribution in different cases: x-axis is the simulation time expressed in seconds, y-axis reports the number of seekers $n_s(t)$ and of parasites $n_p(t)$ in function of time. There are 6 diagrams, and for every diagram the adaptive communication strategy versus the non communicative strategy is reported.

- the population dynamic is depending on the ratio between agents N and food resources M:
    - with scarce resources ($M = 4$) in cases (a),(c),(e) parasites are prevalent
    - with abundant resources ($M = 18$) in cases (d),(f) parasites are balanced with seekers, whereas in case (b) parasites are prevalent.

Table 3.1 resumes the stabilisation property of the population for 3 main cases:

1. silence: agents do not use the food social signal and are still learning.
2. honest: agents signal the food presence honestly and are still learning.
3. dishonest: some agents signals the food presence dishonestly, the rest of them do not signal (like the silence case) and are still learning.

There are 2 important observations to make:

1. the ratio of seekers over parasites (n_s/n_p) is not proportional (directly or inverseley) to the ratio of agents over food sources in the silence and dishonest cases. For example in the silence scenario (agents do not signal the food presence) $n_s = 5, 3, 5$ respectively for $M = 4, 10, 18$. The table shows that the system reaches a stable state but is independent from the proportion.
2. the ratio of seekers over parasites (n_s/n_p) depends proportionally to the ratio of agents to food sources in the honest case. Number of seekers n_s increases accordingly with the food sources M such that $n_s = 4, 8, 10$ respectively for $M = 4, 10, 18$.

Table 3.1: Table summarising self-organisation for 100 simulation runs: every element in the table contains the average number of seekers n_s after stabilisation, the average settling time and the range of the oscillations after the settling time.

Ratio N/M	20:4	20:10	20:18
Silence	5 250 ±2	3 280 ±2	5 156 ±2
Honest	4 390 ±2	8 246 ±2	10 190 ±2
Dishonest agents:2	7 391 ±2	4 249 ±2	8 198 ±2
Dishonest agents:10	5 394 ±2	6 253 ±2	1 202 ±2
Dishonest agents:16	10 410 ±2	3 289 ±2	8 210 ±2

A possible explanation for the prevalence of the parasitic population with scarce resources is due to the space constraints of the food sources: only a few agents (in our case 9) can forage at the same time, therefore the agents learn to transport food for the others. When resources are abundant this constraint is removed therefore the parasitic strategy is no longer needed.

Foraging performance and signalling strategies

If one wants to analyse the performance of the system in consuming food, one could measure the number of the total bites can be measured:

$$F_{tot} = F_{seek} + F_{parasite} \tag{3.41}$$

where F_{seek} is the number of total times agents touched food sources (Eq. 3.18) and $F_{parasite}$ is the number of total times that agents touched other sated agents (Eq. 3.19). Having chosen that index, table 3.2 shows the average F_{tot} of 100 simulations for the different conditions and underlines the value when dishonest is superior to the honest strategy. On the basis of total bites, I can state that in average:

- honest signal is better then silence when food resources are $M = 4, 10, 18$
- comparing the honest and the dishonest strategy: with 10 dishonest agents the F_{tot} is superior to the honest one when $M = 4, 10$ but not when $M = 18$. With only 2 dishonest agents, F_{tot} is bigger only with scarce resources. With 16 dishonest agents, performance is superior only in the intermediate case with $M = 10$ resources.

It is also obvious that the dishonest strategy pays only when resources are scarce, which is also intuitive and has been shown in other simulation or behavioural experiments such as Brembs (1996); Schwieren and Weichselbaumer (2010). The main idea is that if everybody is cheating and is not punished, the information in the system is no more reliable and will punish everybody indiscriminately.

Table 3.2: Table summarising the foraging performance for 100 simulations: every value represents the average F_{tot} as well as the value's range for each group of simulations.

Ratio N/M	20:4	20:10	20:18
Silence	370 ±8	342 ±9	371 ±8
Honest	488 ±5	461 ±6	476 ±4
Dishonest agents:2	$\underline{502}$ ±4	366 ±4	340 ±4
Dishonest agents:10	$\underline{509}$ ±4	$\underline{495}$ ±4	342 ±4
Dishonest agents:16	365 ±4	$\underline{480}$ ±4	362 ±4

Another index for the system performance is the equality in the food distribution. It means that, if I consider the mean μ and standard deviation σ of the food bites over the $N = 20$ agents, the food is better distributed ideally when the average μ is high as well as the deviation σ is high. This indicate that a good distribution is when all agents in average have a good amount of food when the deviation is high and average is high. But if the average is high and the deviation is low, it means that few agents have collected a large amount of food and thus is not desirable for a sustainable society. Table 3.3 contains the computed values for each condition.

- for scarce resources $M = 4$ the best distribution comes with honesty because $\mu_{honest} = 20.00$ and $\sigma_{honest} = 3.0$.

Table 3.3: Table summarising food distribution for 100 simulations: every element represent the couple (μ, σ) where μ is the average of the food bites+agents bites over the agent population, and σ is the standard deviation. They are calculated at the end of each simulation. The max-min range is included for the average of the food bites, as well as the standard deviation.

Ratio N/M	20:4	20:10	20:18
Silence	$(18.65 \pm 0.63, 2.16 \pm 0.77)$	$(17.35 \pm 0.61, 2.83 \pm 0.77)$	$(18.20 \pm 0.64, 3.11 \pm 0.77)$
Honest	$(20.00 \pm 0.43, 3.00 \pm 0.65)$	$(19.50 \pm 0.42, 3.40 \pm 0.65)$	$(20.10 \pm 0.4, 3.30 \pm 0.65)$
Dishonest (2 cheaters)	$(19.40 \pm 0.41, 1.85 \pm 0.66)$	$(17.50 \pm 0.41, 2.09 \pm 0.63)$	$(17.35 \pm 0.41, 3.79 \pm 0.64)$
Dishonest (10 cheaters)	$(19.90 \pm 0.43, 2.13 \pm 0.66)$	$(19.40 \pm 0.44, 2.63 \pm 0.64)$	$(17.10 \pm 0.41, 3.41 \pm 0.63)$
Dishonest (18 cheaters)	$(15.40 \pm 0.40, 5.34 \pm 0.66)$	$(18.65 \pm 0.41, 2.58 \pm 0.63)$	$(17.96 \pm 0.42, 2.67 \pm 0.64)$

- for abundant resources the honest and silence strategies are nearly equal, but with silence the variance is bigger $\sigma_{silence} = 4.11$ vs $\sigma_{honest} = 3.27$.

- for abundant resources the honest strategy has the larger mean $\mu_{honest} = 20.10$ and a comparable variance with the other cases.

Adaptive vs non communicative foraging performance

In this scenario the honest communication strategy is compared to the non communicative behaviour. In figure 3.9 it is reported the the number of total bites F_{tot} (see Eq. 3.41) on the y-axis in function of the simulation time on the x-axis for two cases: adaptive and non communicative strategy when the agents use the social honest signal. In case (a), where resources are scarce, before the sub-systems are formed (before $1x10^4$) F_{tot} is equal in both cases, but during learning and, furthermore when the agents organise, the adaptive case overcomes the non adaptive one. In case (b), where resources are abundant, F_{tot} is equal during all the time, because a strategy to optimize the food gathering is not essential.

Honest behaviour and unlimited resources

In this scenario the agent honestly signals the presence of food places which always produce the G_{food} potential. For this simulation, a population of $N = 20$ agents is provided with $M = 4, 10, 18$ food places sequentially. Population dynamic is observed for a total duration of $T_{sim} = 80000$ (time step $\triangle T = 0.01 seconds$). Figure. 3.10 reports the number of seekers (thick line) and parasites (dotted line) in function of time. The number of seekers $n_s(t)$ is complementary to the number of parasites $n_p(t)$: $n_s(t) + n_p(t) = N$. The population ratios of seeker to parasites $(r(T_{sim}))$ at the end of the simulation for every case $M = 4, 10, 18$ are respectively $r(T_{sim}) = 4/16, 8/12$ and $10/10$.

- in non-adaptive communication: parasites are absent (also with aggressive configuration see 3.28), suggesting that adaptive communication is a necessary condition for the generation of sub-systems.

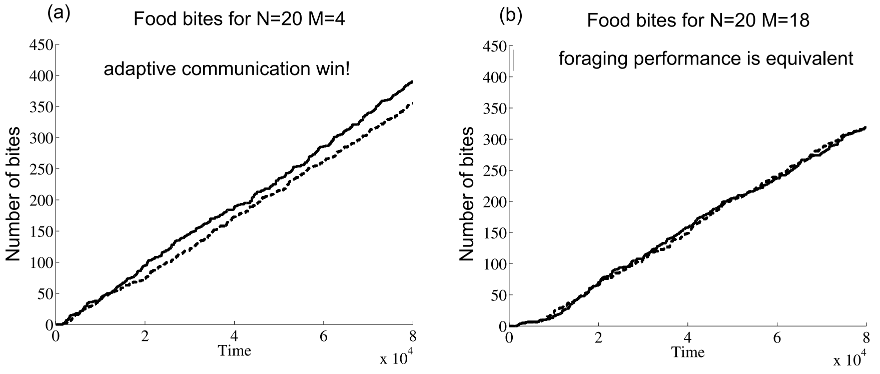


Figure 3.9: Foraging performance comparison: thick line is the adaptive communication with honest signalling and the dotted line is the non-adaptive communication case with honest signalling. **Case (a):** scarce resources adaptive communication win. **Case (b):** abundant resources, performances are equal

- in adaptive communication parasitism is a quasi-stable condition. With scarce resources ($M = 4$), after 600 seconds, the number of seekers (see Fig. 3.10 (a)) stabilises to 4. With abundant resources ($M = 18$), after 600 seconds, the number of seekers (see Fig. 3.10 (b)) stabilises to 10. After 800 seconds (data not shown) small oscillations around the stable point occur in both cases, suggesting that the system has reached an attractor.

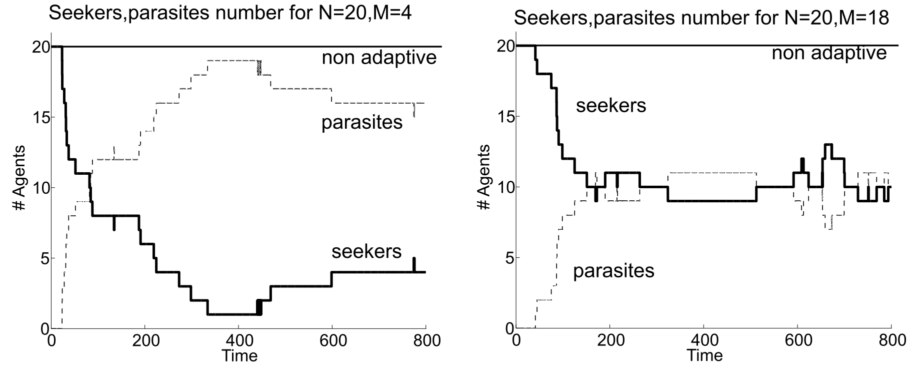
- the population ratio between seekers and parasites $r(T_{sim})$ depends on the ratio between the number of robots and the number of food places N/M:

  – with scarce resources ($M = 4$) parasites are prevalent: $n_p(T_{sim}) = 16 \pm 1 > n_s(T_{sim}) = 4 \pm 1$.

  – with abundant resources ($M = 18$) seekers and parasites are in dynamical equilibrium (oscillate around the stable point after T_{sim} steps): $n_p(T_{sim}) = 10 \pm 1$ and $n_s(T_{sim}) = 10 \pm 1$.

An explanation for the prevalence of the parasitic population with scarce resources is due to the space constraints of the food sources: only a few agents (in our case 9) can forage at the same time, therefore agents ”transport” food for the others. When resources are abundant this constraint is removed therefore parasitism is not essential.

To analyse the performance of the system in consuming food: the number of the total bites $F_{tot} = F_{seek} + F_{parasite}$ it is considered. F_{seek} is the number of total times that agents touched food sources (Eq. 3.18) and $F_{parasite}$ is the number of total times that agents touched other sated agents (Eq. 3.19). In table 3.4 it is reported the foraging performance (average ± range) over 100 simulations and it can be noticed that adaptive communication provides the best performance only for scarce resources and an equal performance for abundant resources.



(a) Case considering scarce resources: in non adaptive-communication only seekers are present (top line is constant to 20 agents), in adaptive communication sub-system formation is achieved and parasites become more than seekers after 80 seconds

(b) Case considering abundant resources: in non adaptive-communication only seekers are present (top line is constant to 20 agents), in adaptive communication sub-system formation is achieved and parasites are in equilibrium with seekers after 200 seconds

Figure 3.10: A typical single run of the system which shows different cases.

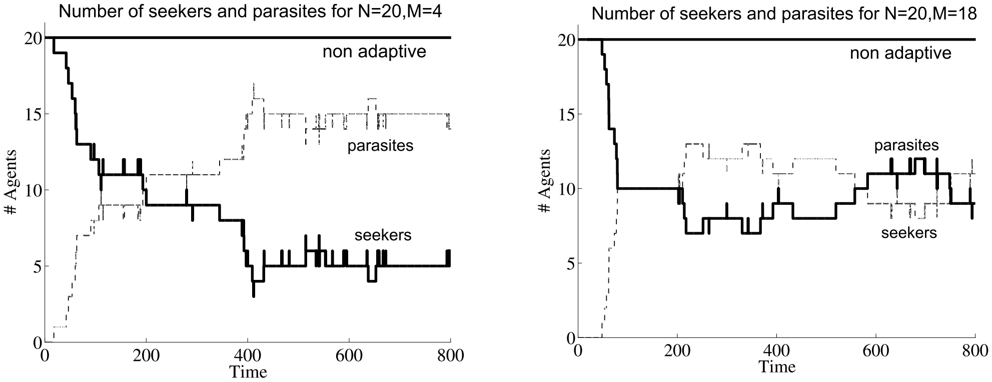
Table 3.4: Table summarising foraging performance over 100 simulations. With scarce resources, performance is superior using adaptive communication, with abundant resources performances are equal since food distribution through parasites is not essential.

Ratio N/M	20:4	20:18
Adaptive communication	$F_{tot} = 396 \pm 2$	$F_{tot} = 319 \pm 2$
Non-adaptive communication	$F_{tot} = 355 \pm 2$	$F_{tot} = 319 \pm 2$

Honest behaviour and limited resources

In this scenario the agent honestly signals the presence of food which is limited in time. For this simulation, a population of $N = 20$ agents is provided with $M = 4, 10, 18$ food places sequentially. Population dynamic is observed for a total duration of $T_{sim} = 80000$ (time step $\triangle T = 0.01 seconds$). The population ratios of seeker to parasites $(r(T_{sim}))$ at the end of the simulation for every case $M = 4, 10, 18$ are respectively $r(T_{sim}) = 5/15, 8/12$ and 10/10.

- in non-adaptive communication: parasites are absent (also with aggressive configuration see 3.28), suggesting that adaptive communication is a necessary condition for the generation of sub-systems.
- in adaptive communication parasitism is a quasi-stable condition. With scarce resources ($M = 4$), after 600 seconds, the number of seekers (see Fig. 3.11 (a)) stabilize to 5. With abundant resources ($M = 18$), after 600 seconds, the number of seekers (see Fig. 3.11 (b)) stabilises to 10. After 800 seconds (data not shown) small oscillations around the stable point occur in both cases, suggesting that the system has reached an attractor.
- the population ratio between seekers and parasites $r(T_{sim})$ depends on the ratio between



(a) Case considering scarce resources: in non adaptive-communication only seekers are present (top line is constant to 20 agents), in adaptive communication sub-system formation is achieved and parasites become more than seekers after 200 seconds

(b) Case considering abundant resources: in non adaptive-communication only seekers are present (top line is constant to 20 agents), in adaptive communication sub-system formation is achieved and parasites are in equilibrium with seekers after 200 seconds

Figure 3.11: A typical single run of the system which shows different cases: self organisation with honest behaviour and limited resources

the number of robots and the number of food sources N/M:

- with scarce resources ($M = 4$) parasites are prevalent: $n_p(T_{sim}) = 15 \pm 1 > n_s(T_{sim}) = 5 \pm 1$.
- with abundant resources ($M = 18$) seekers and parasites are in dynamical equilibrium (oscillate around the stable point after T_{sim} steps): $n_p(T_{sim}) = 10 \pm 2$ and $n_s(T_{sim}) = 10 \pm 2$.

An explanation for the prevalence of the parasitic population with scarce resources is due to the space constraints of the food places: only few agents (in our case 9) can forage at the same time, therefore agents ”transport” food for the others. When resources are abundant this constraint is removed therefore parasitism is not essential.

To analyse the performance of the system in consuming food: the number of the total bites $F_{tot} = F_{seek} + F_{parasite}$ it is considered. F_{seek} is the number of total times that agents touched food sources (Eq. 3.18) and $F_{parasite}$ is the number of total times that agents touched other sated agents (Eq. 3.19). In table 3.5 the foraging performance (average ± range) over 100 simulations is demonstrated and it can be noticed that adaptive communication provide the best performance in term of food foraging for scarce and abundant resources. Why is the performance better than the unlimited case? Because this time the food resources are limited in time and space, so even when there are a lot of food resources the agents must collaborate to maximise the food collection.

Table 3.5: Table summarising foraging performance over 100 simulations. Adaptive communication has the best performance for both scarce and abundant resources.

Ratio N/M	20:4	20:18
Adaptive communication	$F_{tot} = 439 \pm 2$	$F_{tot} = 340 \pm 2$
Non-adaptive communication	$F_{tot} = 320 \pm 2$	$F_{tot} = 230 \pm 2$

Stability analysis for changes in population

Every self organizing system should be stable to external variations in the parameter space, this property is generally known as robustness to variation. If the number of agents are changed during simulation the system reacts promptly and stabilize to a new state. In Fig.3.12 seekers are stabilised in the range 4 ± 1, when 6 more agents are added at time 400, seekers stabilize again in the range 7 ± 1. Table 3.6 contains a summary of the system reaction to different sets of disturbances for 100 simulations.

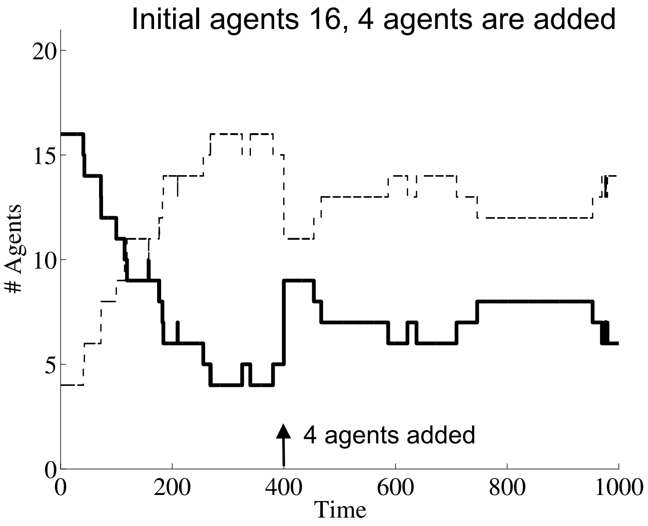


Figure 3.12: Simulation starts with 16 agents, then 4 agents are added after 400 seconds.

This property indicates that the system is robust to a variation in the number of agents: the system will adjust automatically to the new equilibrium with a certain response time. The system is also robust to a variation in the food resources, although with a slower response time (not shown in the Figure). This stability is a good feature especially for real implementation of social systems where disturbances from the environment are likely to occur very often and the designer is certain that the system will be stable enough to keep with the foraging task.

Table 3.6: Table summarising system stability to different disturbances for 100 simulation runs. Each cell contains the settling time and the number of seekers after the addition of n agents at time step 400.

Ratio N/M	20:4	20:10	20:18
After adding 4 agents	50 3 ±2	50 6 ±2	50 7 ±2
After removing 4 agents	65 7 ±2	65 9 ±2	65 10 ±2
After adding 10 agents	76 2 ±2	76 3 ±2	76 5 ±2

3.1.10 Discussion: sub system formation in the social model

The simulations of a social system based on pro-active learning agents has shown very promising results. First of all the system self-organizes by means of individual specialization to the environment conditions which are the number of agents and number of food resources. Agents are based on a simple, yet effective learning mechanism called ICO which can be extended to implement competitive behaviours. Because internal processing is avoided, agents respond to the changes of their environment in a timely fashion. The system is able to generate subsystems as Luhmann proposed by means of communication: the agents reduces the complexity of the environment by selecting only one behaviour out of two possiblee ones. This selection is mediated by the communication of the food status which allows the agent to make a choice or selection. By doing so every group of agents is more predictable in respect to each other thus generating a self-organizing collective behaviour which can be identified by the two subsystems. Other embodiments for the learning controller could have been chosen, for instance with Q-learning (Watkins and Dayan, 1992) reactive agents are given a description of the current state and have to choose the next action so as to maximise a scalar reinforcement received after each action. The task of the agent is to learn from indirect, delayed reward, to choose sequences of actions that produce the greatest cumulative rewards. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states. In economics and game theory, reinforcement learning is considered as a boundedly rational interpretation of how equilibrium may arise. Reinforcement models for MAS suffers of 2 disadvantages:

1. complexity may be exponential in the number of environmental states.
2. discrete models: agents choose from a set of actions in a discretised world

This problem is crucial in MAS: when an agent is learning the value of its actions in the presence of other agents, it is learning in a non stationary environment. In Haitao et al. (2000) a novel exploration strategy, for the Q-algorithm applied in a predator-prey game to obtain convergence, is introduced. Unfortunately the cost of communication in their model is very expensive, so they discuss only the case without communication. Our learning rule ICO, is unsupervised and is computationally efficient (since it does not rely on states) and inspired on the evidence that organism tends to maintain a weak homoeostasis with the environment as demonstrated by McFarland (1993). However some researchers argue that also ICO learning is a simplified form of Q-learning by identifying the reward signal as the reflex of the organism:

the biological difference here is that the reflex reward is something evolutionary wired whereas the reward is usually artificially encoded in the environment. Therefore we could say that also ICO learning is also supervised but more biologically plausible. In Tan (1997) Q-learning is applied in a predator-prey game, where agents cooperate in different ways. It is interesting to analyse the communication method called sharing sensation. The model is composed of 1 hunter, 1 prey and a scouting agent. At each step, the scout sends its action and sensation back to the hunter: the hunter relies initially on his sensation and then on the scout's sensation. Therefore the scout can be compared in our model as the distal signal: hunter can see the prey at longer distances. Performance (number of steps to capture a prey) is then compared in the 2 cases: scouting versus no scouting. Performance is superior in the scouting case ,as well as in our model performance, it is superior when agents use both proximal and distal signal (see table 3.4). Another approach used to develop communication in MAS is in Floreano et al. (2007) where a genetic-algorithm is used to evolve 1000 robots (divided in 100 colonies). The control system used is a feed-forward neural network with 10 inputs and 3 output neurons. The network was encoded using a genetic string of 240 bits. Synaptic weights are only evolved and not learned and robots had a sensory-motor cycle of 50 ms. Our controller makes use of 12 inputs (2 more) and 2 motor neurons + 1 the $G_{satedness}$ signal, but does not use a sensory-motor cycle allowing fast responses. Thus Floreano et al. (2007) makes use of genetic selection and recombination to produce robots behaviours (communication strategies). Hence the agent system do not self organise in classes. Other similar works, making use of evolved recurrent neural networks (RNNs) are reported in Wischmann and Pasemann (2006) and Marrocco and Nolfi (2006). In Marrocco and Nolfi (2006) a population of agents evolved for the ability to solve a collective navigation problem develop individual and social/communication skills. A particular evolved behaviour resembles our system differentiation: "a differentiation of the modalities with which communication is regulated (... e.g. specialised asymmetrical interaction forms in which one robot acts as a speaker and one robot acts as an hearer)". In Wischmann and Pasemann (2006) the evolutionary adaptivity of RNNs to varying environmental conditions, such as the number of interacting robots is studied. In my model the system specializes in a group that gets the food and distributes it (seekers) and the other one (parasites) collects it. The communication strategy was robust only for small changes. In our model, agents adapt continuously to the environment, they self-organise efficiently with varying robot number N and varying M food sources. Moreover the system converges to a quasi-stable state thanks to the stability provided by the learning rule. The system performance in terms of food foraging behaviour was analysed for a honest and dishonest signalling strategy both in terms of food distribution and in terms of total energy acquired. This study can be used to predict the performance of a social robotic system in a real test case so that the designer can choose the parameters according to the desired performance. In the following sections, I will introduce input measures that quantify the agent performance and information selection as suggested by Luhmann.

3.2 An input based information measure for adaptive controllers

This Section describes the development of an information measure to assess the learning performance of the, previously described, learning agents in the social system.

Therefore the main purpose of this Section is to introduce a couple of correlation based performance measures which are computed at the sensory inputs of the agent. The first measure is called **maxcorr**, is very simple and can be computed in real time without any probability estimation. The **Maxcorr** measure can then be normalised by using a logarithmic approach to express the learning performance in terms of bits and thus is called **AI measure** . The simplicity of the AI measure derives from the dependency on the representation used by the controller/agent, but it has the disadvantage of not being general, as described in the discussion.

3.2.1 Introduction to the maxcorr input measure

This section describes an information measure suitable for closed loop controllers that makes use of temporal unsupervised learning. It can be applied for example to ICO/ISO learning, differential and temporal Hebbian learning. This information measure estimates how much the agent/controller has learnt and is directly correlated to the weight change of the learning rule used. The measure is based on the agent's perspective (at the input side) rather than at output side as previous approaches. Looking at the output is equivalent to analysing behaviour and has a disadvantage: agents adapt to the complexity of the environment thus even a simple reactive agent has increasing complex behaviour in an increasingly complex environment (Nolfi, 2005). As I argued before in the previous Section 2.5, organisms perform input control because neural activity (motor output) generates other neural activity (sensory input) and so it is clear that an input measure can capture the performance of any agent, especially the one that are based on ICO/ISO controllers which performs input correlation. Input control is a closed loop property and is agent centric, whereas output control is observer centric. To my best knowledge, this is the first approach for the closed loop case, which is not based on the traditional concept of Shannon's entropy. In this model agents are learning to avoid obstacles and each other using a proximal signal (touch sensor as pain) and a predictive signal (vision). An information measure is computed on these inputs to verify if the agents are learning. Firstly the author introduces the information measure based on the cross correlation of the agent inputs, then he applies the measure to two types of agent controllers (a complex and a simplified model) for an avoidance task with multiple agents. Finally the measure is computed on the social setting described in section 3.1.

3.2.2 Methods: controller assumptions

The following assumptions were made for the development of the measure:

- the controller is a MIMO (multi input multi output) or MISO (multi input single output)
- input signals can be analog or digital

The closed loop measure is based on the cross correlation to be applied to the input signals (minimum is 2) of the adaptive controller. Cross correlation of two real analog signals $x(t), y(t)$ of a real variable t (in our case time), denoted as $corr(x, y)$ is defined by:

$$corr(x, y, t) = \int_{-\infty}^{\infty} x(t + \tau), y(\tau) d\tau \tag{3.42}$$

Cross correlation for realisable devices (software or hardware implementations) must be computed in finite time, that means computing the cross correlation in a limited time window

$T < \infty$:

$$corr(x, y, t) = \int_{t}^{t+T} x(t + \tau), y(\tau) d\tau \tag{3.43}$$

Cross correlation for digital signals (sampled analog signals) as two discrete time series x_n, y_n with $n = 1, .., N$ is defined as:

$$R_{xy}(m) = \sum_{n=0}^{N-m-1} x_{n+m} y_n \qquad m \geq 0 \tag{3.44}$$

$$R_{yx}(-m) = 0 \qquad m < 0 \tag{3.45}$$

$$c_{xy}(m) = R_{xy}(m - N) \quad m = 1, 2, ..., 2N - 1 \tag{3.46}$$

The cross correlation give a measure of the linear synchronisation between x and y. Its absolute values are normalised in the range from zero (no synchronisation) to 1 (maximum synchronisation). The cross correlation is symmetric: $c_{xy}(m) = c_{yx}(m)$. Peaks in the cross correlation signal determine the phase delay between signals (for other cross-correlation variation see 5.5.1).

The next section describes a complex model for an avoiding robot using the ICO learning rule.

3.2.3 Methods: complex model configuration

The simulation model is composed of a 2 dimensional world bounded by walls as described before in Chapter 3.1.1. The agents are indexed by their position as $a_j(t)$, where a_j has 2 components (x,y coordinates indexed by $a_{j,x}$ and $a_{j,y}$), with $j = 1, .., N$. Agents move with a differential drive system (two wheels).

In this configuration the agent has only an avoidance behaviour:

- left and right reflexes makes him retract when both synapses are active
- left and right distals are being used later on when the agent has learned the association of distal to reflexes.

Agents and walls are obstacles. Agents produce obstacle signals. Every agent $a_j(t)$ has a potential field associated (see Eq.3.1):

$$Gavoid_j(t) = G(x - a_{j,x}(t), y - a_{j,y}(t)). \tag{3.47}$$

that is sensed by the corresponding inputs of other agents $a_k(t)$ (with $k \neq j$) labelled as $avoid_{left,right}$. Walls don't produce G_{avoid}, so that agents sense them using proximal signals that are generated by collisions: when 2 distinct agents j and l collide at time t_0, such that $||a_j(t) - a_l(t)||_2 < D$ (D is the radius of the agent) an impulse is produced at $avoid_{l,r}$. The agent's controller, (shown in Fig. 3.13) is composed of:

- two input synapses: left and right with one reflex and one distal signal per synapse.
- two ICO neurons: left and right cross connected as left synapse with right motor and right synapse to the left motor.

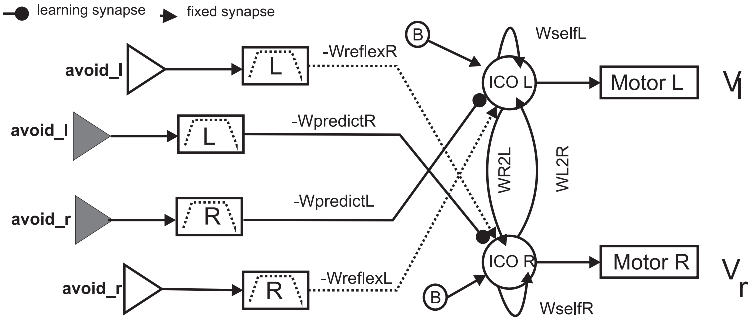


Figure 3.13: Avoidance network: ICO left and right are two neurons implementing the ICO learning rule, their output is a sigmoid and is connected (after normalisation into $[v_{min}, v_{max}]$) to the motor speed commands. Grey triangles represent distal inputs, while white triangles represent proximal inputs. The learned synaptic weights are associated to the distal synapses (thick lines) while the fixed are associated to the proximal synapses (dotted lines). To produce a retraction behaviour left and right weights must be different such that $W_{predict,L} > W_{predict,R}$ and $W_{reflex,L} > W_{reflex,R}$, if $W_{predict,L} = W_{predict,R}$ and $W_{reflex,L} = W_{reflex,R}$ robot will just go back without turning.

- two output motors: respectively for the left and right ICO neuron connected to the left and right motor speed.

Every ICO neuron (left and right) has 2 corresponding reflexes (left, right short range sensors) connected:

$$x_{0,l}(t) = avoid_{prox,r}(t) \quad x_{0,r}(t) = avoid_{prox,l}(t) \tag{3.48}$$

and 2 corresponding predictive signals (left, right long range sensors) connected:

$$x_{1,l}(t) = avoid_{dist,r,d_o}(t) \quad x_{1,r}(t) = avoid_{dist,l,d_o}(t). \tag{3.49}$$

The neural controller for the avoidance behaviour is shown in Fig. 3.3 (Stamm, 2006), every ICO neuron (left and right) computes the following operations:

$$\begin{aligned} ICO_L(t) &= B - h * x_{0,r} \cdot W_{reflexL} - h * x_{1,r} \cdot W_{predictL} & (3.50)\\ &+W_{selfL} \cdot ICO_L(t-1) + W_{L2R} \cdot ICO_R(t) \\ ICO_R(t) &= B - h * x_{0,l} \cdot W_{reflexR} - h * x_{1,l} \cdot W_{predictR} & (3.51)\\ &+W_{selfR} \cdot ICO_R(t-1) + W_{R2} \cdot ICO_L(t) \end{aligned}$$

The parameters used for the weights, the bias and the recurrent connections are reported in the Appendix sections 5.2.4,5.2. The recurrent connections between the left and right ICO neuron, are necessary to implement a push-pull behaviour so that when the robot synchronously activates both the left and the right input, only one ICO neuron will dominate thus evoking a turn-back response. Connections between input synapses and ICO neurons (motor neurons) are negative to evoke a retraction. The motor output is calculated with a sigmoid activation function on the ICO neuron membrane as follow:

$$V_L = \frac{1}{1+e^{-ICO_L}} \tag{3.52}$$

$$V_R = \frac{1}{1+e^{-ICO_R}} \tag{3.53}$$

The weight update learning rule is calculated for the weights:

$$\frac{\partial W_{predict,L}}{\partial t} = \mu \cdot x_{1,l} \frac{\partial x_{0,l}}{\partial t} \tag{3.54}$$

$$\frac{\partial W_{predict,R}}{\partial t} = \mu \cdot x_{1,r} \frac{\partial x_{0,r}}{\partial t} \tag{3.55}$$

3.2.4 Methods: cross correlation and ICO learning

Cross correlations are computed for the left and right synapse at fixed non overlapped time windows $\triangle T$ expressed in seconds. Considering Eq.3.46 $N_s = \triangle T/\delta t$ is the number of samples in a time window of $\triangle T$ with a δt as sampling time. Left and right cross correlations are computed for every non overlapping time window $k = 1, 2, ...$

$$xcorr_{left}(k) = corr(H(x_{1,l}), H(x_{0,l})) = corr(u_{1,l}, u_{0,l}) \tag{3.56}$$

$$xcorr_{right}(k) = corr(H(x_{1,r}), H(x_{0,r})) = corr(u_{1,r}, u_{0,r}) \tag{3.57}$$

where k is the index of the time window where the cross correlation was computed, H is a high pass filter that removes the constant component (generally referred as DC) from the cross correlation diagram. Removing the DC bias is necessary for the application of statistical measures (see Appendix 5.5.1). In my the bias removal is already taken care by the high pass filters as shown in Figure 3.13. For every time window k, I computed the maximum max, energy E,power P of the left $xcorr_{left}(m, k)$ and right $xcorr_{right}(m, k)$ cross correlation plus the average of the weight change for left and right $W_{predict,L}, W_{predict,R}$.

$$M(xcorr_{left}(k)) = \max_m(xcorr_{left}(m, k)) \tag{3.58}$$

$$M(xcorr_{right}(k)) = \max_m(xcorr_{right}(m, k)) \tag{3.59}$$

$$E(xcorr_{left}(k)) = \sum_m xcorr_{left}(m, k)^2 \tag{3.60}$$

$$E(xcorr_{right}(k)) = \sum_m xcorr_{right}(m, k)^2 \tag{3.61}$$

$$P(xcorr_{left}(k)) = \frac{\sum_m xcorr_{left}(m,k)^2}{N} \tag{3.62}$$

$$P(xcorr_{right}(k)) = \frac{\sum_m xcorr_{right}(m,k)^2}{N} \tag{3.63}$$

$$Avg(W_{predict,left}, k) = \frac{\sum_m W_{predict,left}(m+kN_s)}{N_s} \tag{3.64}$$

$$Avg(W_{predict,right}, k) = \frac{\sum_m W_{predict,right}(m+kN_s)}{N_s} \tag{3.65}$$

The reason why I have computed this values for the left and right synapse is that these weights develop independently from each other as in Eq. 3.54,3.55. Since energy and power are related measures only energy is computed (see Appendix 5.5.3). The average of the weight change for the left and right distal synapses is computed to validate the information measure for the result section.

3.2.5 Methods: a simplified model

The agent controller can be over simplified by using only one weight and thus one ICO controller for the avoidance behaviour. The difference between the left and right far antennas provides x_1 and the difference between the left and right near short antennas provides x_0. The band pass filters in Fig.3.14 generate u_0, u_1 that are damped waves if x_0, x_1 are delta pulses. The transfer function of the band pass filter is specified in the Laplace-domain as:

$$\begin{aligned} h(t) &\leftrightarrow H(s) = \frac{1}{(s+p)(s+p*)} && (3.66) \\ h(t) &= \frac{1}{b} e^{at} sin(bt) && (3.67) \\ a &= -\pi \frac{f}{q} && (3.68) \\ b &= \sqrt{(2\pi f)^2 - a^2} && (3.69) \end{aligned}$$

where $p*$ represents the complex conjugate of the pole $p = a + ib$, f is the oscillation frequency and q is the quality factor of the filter. ICO correlates the predictive signal u_1 with the reflexive signal u_0 according to the formula:

$$\frac{d\omega_1}{dt} = \mu \cdot u_1 \cdot \frac{du_0}{dt} \qquad (3.70)$$

where ω_1 functions now as the weights of the more complex model in Eq. 3.54,3.55. Then the output z of the controller is used to control the steering angle of the robot such that an obstacle on the left $u_0 > 0$ will produce an anticlockwise turn, whereas an obstacle on the right $u_0 < 0$ will produce a clockwise turn. The controller learns to avoid the error signal u_0 using the predictive signal u_1. Fig.3.14(A) illustrates how the learning is achieved and (B) describes how the agents interact with the world. A purely reactive agent has only a reflexive behaviour via u_0 and will never learn to avoid the loop error signal u_0: it will touch the obstacle and produce a trajectory like (1). When the agent starts to learn ($\omega_1 > 0$) it will use the u_1 to prevent u_0, thus avoiding the obstacle before touching it like the trajectory in (2). Figure 3.14(C) shows how the reflex signal is shifted forward in time and reduced in amplitude due to the anticipatory motor reaction of the controller.

3.2.6 Methods: anticipatory information

Within the simplified model, the equations in 3.58 and 3.59 collapse in a unique value which is defined as $cc(t)$ as in Eq. 3.71. Intuitively the information measure grows when the agent is using anticipatory information and is zero when the agent is not able to predict its reflex. Before learning the predictive signal u_1 is followed always by the reflex signal u_0 with the same

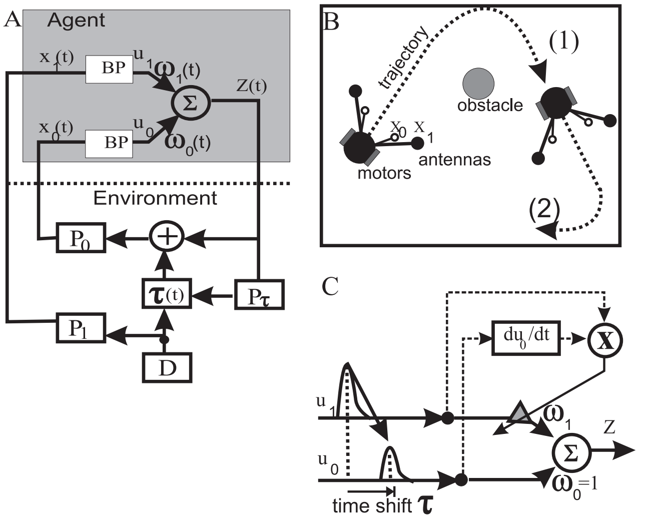


Figure 3.14: A) Schematic diagram of the closed-loop learning system with inputs x_0 and x_1, synaptic weights ω_0 and ω_1 and motor output z. P_0 and P_1 are the transfer functions of the reflexive and the predictive pathway. BP block is a 2 pole band pass filter. B) Agent setup with short range antennas (reflexive inputs, x_0) and long range antennas (predictive inputs, x_1). The agent is learning to avoid obstacles and walls using its short and long range antennas. The motor reaction will reduce the intensity of the painful reflex x_0 as well as delay its occurrence. C) Schematic diagram of the input correlation learning rule and the signal structure (Porr and Wörgötter, 2006). The u_0 and u_1 are respectively the difference between the filtered values of the left and right antennas of the agent. During learning the u_0 peak will be shifted in time and reduced in amplitude.

amplitude, while after learning the amplitude of u_0 is likely reduced. An ideal learner will be able to reduce totally the reflex u_0 to 0. One will also notice that generally, after learning, the reflex u_0 will not only be reduced but also delayed in time.

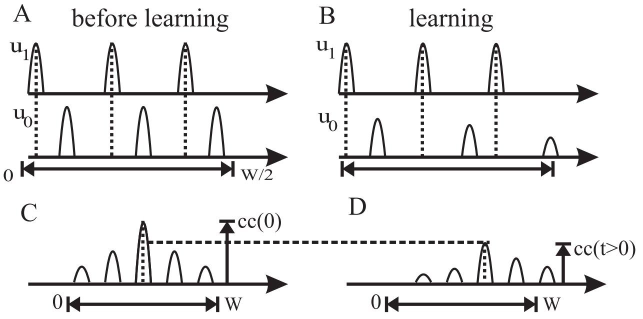


Figure 3.15: A) Illustration of the signals u_1, u_0 of a non learning agent. The peaks are periodic but off phase and with same amplitude. B) Same time diagram for a learning agent. C) Cross correlation of u_1 and u_0 for the non learning case. D) Cross correlation of u_1 and u_0 for the learning case.

Fig.3.15 shows a typical temporal diagram of events for a non learning agent A and for a learning one B. If one takes the cross correlation between the u_1 and u_0 for both cases, it is possible to understand by the cross correlation which agent is learning and which one is not. For instance the non learning agent has a cross correlation that is not shifted in time and reduced in amplitude like in Fig.3.15(C), while a learning agent has a cross correlation that is shifted in time and reduced in amplitude like in Fig.3.15(D). My purpose is now to quantify this performance by using an information based measure in terms of information bits. Therefore a normalised version of $cc(t)$,AI is computed in 2 steps:

$$cc(t) = max(\sum_{\tau=-\frac{W}{2}}^{\tau=\frac{W}{2}} u_1(t) \cdot u_0(t+\tau)) \tag{3.71}$$

$$AI(t) = -log_2(\frac{cc(t)}{cc(0)}) \tag{3.72}$$

$$0 \leq \frac{cc(t)}{cc(0)} \leq 1 \tag{3.73}$$

$cc(t)$ is the maximum of the cross correlation between the error signal and the predictive signal in the time window W that must be sufficiently large to take at least a pairing of u_1 with u_0 when learning is off. $cc(0)$ is the maximum of the cross correlation computed when the agent is not learning, whereas $AI(t)$ takes the ratio between the current and the initial cross correlation, thus the argument of the logarithm will range from 0 to 1 because the correlations following the first one can at least be equal to the first one. When learning is off, the agent's predictive signal precedes the reflex signal whose amplitude is not reduced hence $AI \simeq 0$, when learning is on, the agent learns to reduce the error u_0, using an earlier motor reaction elicited by u_1, thus the $AI \to \infty$ in the ideal case of perfect learning. In terms of information bit, a reduction of $cc(t)$ by half can be interpreted as an improvement of 1 bit as discussed in the results section. In terms of information, if the agent is learning continuously the number of bits of the AI will increase in time until the agent has completely avoided the reflex. In the next section I go back to the complex model and compute the cross correlation and energy for the left and right synapses. After that I will perform a benchmark of AI for a the simplified model. Both measures are taken in a multi agent scenario to see the effects of multiple learning agents. I then compute the AI for the simplified model in the social system and draw some conclusions.

3.2.7 Results: complex model results

Simulations with the complex model were executed with an increasing number of agents in a rectangular two-dimensional world with obstacles. The software used to simulate the agents is Enki an open source simulator for multiple robots interacting on a flat surface. The simulator implements collisions, physics support (like slip, friction etc..) and features 4 realistic robots. For our simulations I used a group of Alice robots and setup the experiment as follow:

- $N = 2$ agents and $M = 2$ obstacles
- $N = 4$ agents and $M = 2$ obstacles
- $N = 4$ agents and $M = 2$ obstacles, and an introduction of an agent

Agents for every case are numbered from 0 to $N - 1$. The world's area A_{world} is proportional of a factor K to the sum of the agent's area:

$$A_{world} = K_a \cdot (\sum_{i=1}^{N} A_{agent}) + K_o \cdot (\sum_{i=1}^{M} A_{obstacle}) \tag{3.74}$$

where A_{world},A_{agent},$A_{obstacle}$ are respectively the area of the world, the area of the agent and the area of the obstacle. This normalisation technique is necessary to provide approximately the same amount of sensory events when the area get more crowded. Intuitively speaking, if I keep the same area and add an increasing amount of agents there will be a proliferation of collision and thus reflexes as well as predictor events and thus comparison of the measures will not be reliable. Simulation time for every setup was set to $T_{max} = 600000$ steps, $\triangle T = 600$, with sampling time $\delta t = 0.01$ seconds. Learning is switched on $\mu > 0$ for $t > \triangle T = 600$: the cross correlation of the first time window is computed when agents are only using reflexes (reactive agents). If I compute the measure for two purely reactive agents as in Fig. 3.16, I can see that the values are constant for the left and right synapse: the agent is not learning anything about the causal relations of distal and proximal signal. This is because the $u1$ event is followed always by a $u0$ with the same amplitude and thus $xcorr$ amplitude is steady as described in the simplified model of Figure 3.15(A,C).

Another important property to notice in Fig. 3.16 is the difference in the offset between the $M(xcorr_{left}(k))$ left maximum cross correlation and the $M(xcorr_{right}(k))$ right maximum cross correlation. This is due to the initial bias of the avoiding behaviour for the agent, because $W_{predict,L} > W_{predict,R}$ and thus the agent tends to turn on the left each time an obstacle is encountered.

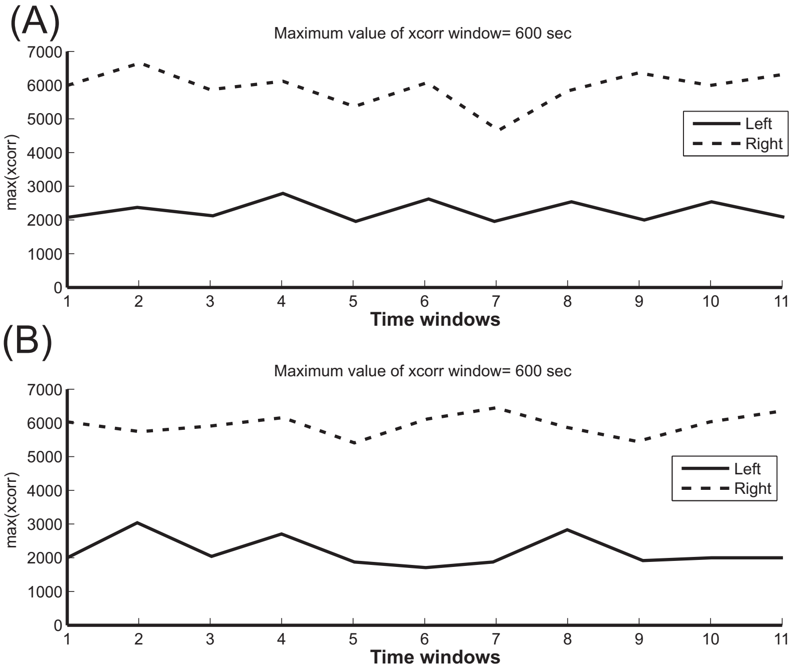


Figure 3.16: Two agents are moving in the environment without learning $\mu = 0$. The $M(xcorr_{left}(k)), M(xcorr_{right}(k))$ are computed for each reactive agent, A and B. The measure is constant: agents are not learning anything from the environment.

Experiment with 2 agents and 2 obstacles Figures 3.17,3.18 show the measures for $N = 2$ learning agents and $M = 2$ obstacles.

- Fig.3.17(a) contains the $M(xcorr_{left}(k))$, $M(xcorr_{right}(k))$ in the upper panel and the $Avg(W_{predict,left}, k)$, $W_{predict,right}, k)$ in the lower panel for the first agent.
- Fig.3.17(b) contains the $M(xcorr_{left}(k))$, $M(xcorr_{right}(k))$ in the upper panel and the $Avg(W_{predict,left}, k)$, $W_{predict,right}, k)$ in the lower panel for the second agent.
- Fig.3.18(a) contains the $E(xcorr_{left}(k), E(xcorr_{right}(k))$ in the upper panel and the $Avg(W_{predict,left}, k)$, $W_{predict,right}, k)$ in the lower panel for the first agent.
- Fig.3.18(b) contains the $E(xcorr_{left}(k), E(xcorr_{right}(k))$ in the upper panel and the $Avg(W_{predict,left}, k)$, $W_{predict,right}, k)$ in the lower panel for the second agent.

Each agent this time is learning $\mu > 0$ to avoid each other and so the weights are decreasing $W_{predict,right}, k)$ like in Fig.3.17(a) bottom, although not equally because the left input is stronger and does not allow the same amount of learning on the right input. The weight change reflects the corresponding correlation measure Fig.3.17(a) up, because the cross correlation for the left $M(xcorr_{left}(k)) = 0$ and right $M(xcorr_{right}(k)) = 0$ goes to zero when the weights are stabilised, indicating that the reflex is not triggered any more when $k > 7$.

Oscillation of $M(xcorr_{left,right})$ are present because the agent is learning on average to avoid the reflex: that means sometimes due to the complexity of the environment some unpredictable events can still occur. For example agent 1 can appear suddenly in the range of agent 0 which is already avoiding another obstacle and inevitably will hit either the obstacle or the agent 1.

The energy of the signal is useful to estimate the complexity of the environment composed by the obstacles and the other learning agents: the more pairing of distal and proximal events the bigger the energy. Energy is high $E(xcorr_{left}(k))$ and $E(xcorr_{right}(k))$ when all agents are unpredictable (learning rate is not stable) $k < 6$ as in Fig.3.17(b),3.18(b) Indeed increasing the number of agents to $N = 4$ increases the level of energy and conversely, decreasing the number of agents reduces the level of energy.

Experiment with 4 agents and 2 obstacles The same considerations apply to the case of 4 agents with 2 obstacles. Figs.3.19,3.20 shows the measures for the first group of 2 agents and Figs.3.21,3.22 shows the measures for the second group of 2 agents.

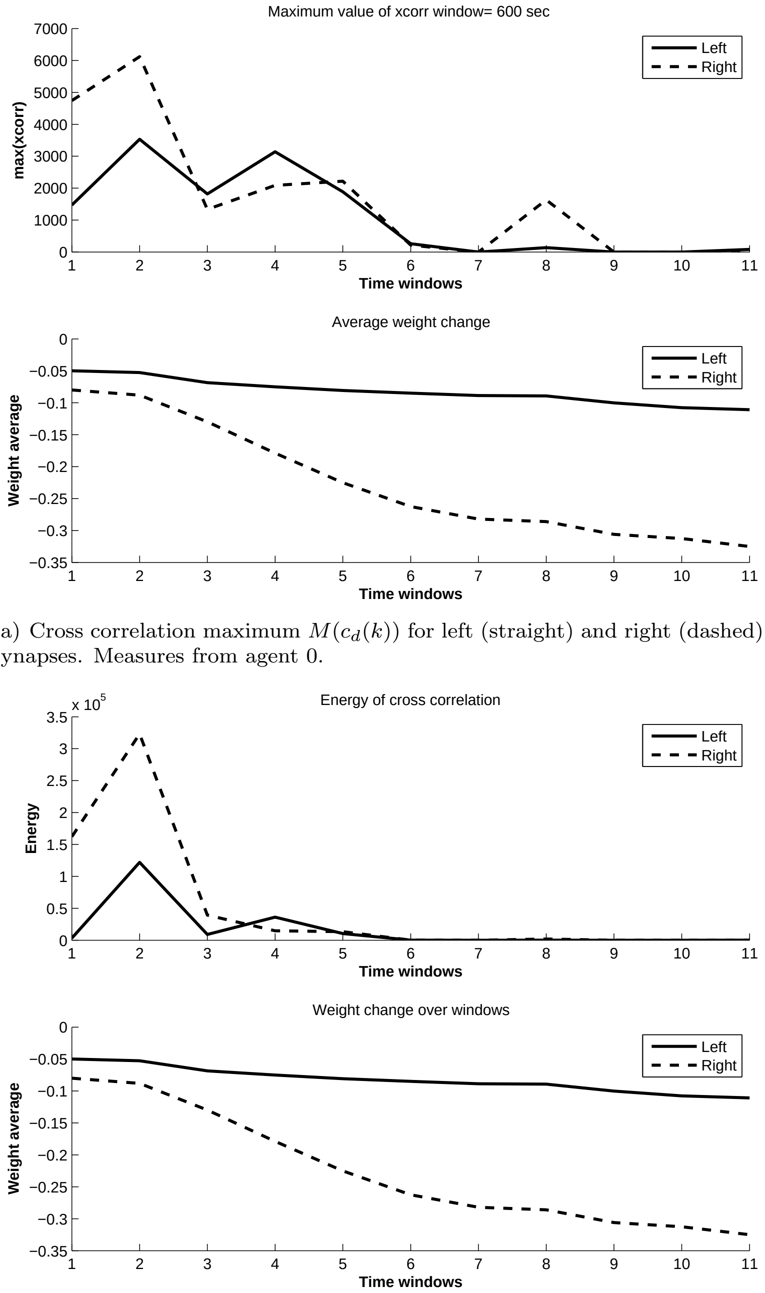
As with the previous case the maximum cross correlations of each agent go to zero when the learning is stable. For example agent 2 in Fig.3.19(a,b) stabilises his learning weights after $k > 6$ time windows. Comparing the maximum of the cross correlation for left and right synapses for all cases, on average it occurs that $M(xcorr_{left}(k)) > M(xcorr_{right}(k)),\forall k$: agents learn more to use the left synapse which therefore is more responsible for the motor behaviour. As a result the left synapse is more exposed to obstacle signals. Instead the agent in Fig.3.21(a) learns equally with both synapses and in this case $M(xcorr_{left}(k)) \cong M(xcorr_{right}(k))$.

An interesting property of the energy $E(xcorr_d(k))$ is that in all cases it tends to zero $E(xcorr_d(k)) \cong 0$ when the weight change is stable $Avg(W_{predict,d}, k) \cong 0$. But there are exceptions like cases in Fig.3.20(b),3.22(b) where $E(xcorr_d(k)) \cong 0$ for $k > 5$ even if the agent is still slowly learning. This is due to the fine tuning that happens after the initial learning step, essentially there are still some minor weak collisions which adjust the weights by a small factor.

The energy values can be used to for example agent 2 has a peak in the energy value as in Fig.3.21(b) because there was a great learning experience at $k = 4$, where the weights' values jumped from -0.08 to -0.18. Because the energy value calculates the density of collision events, it can be used to verify the learning speed of the weight development.

Adding one agent When one more agent is added in the simulation at the time window $k = 6s$, the other agents are "surprised" by its behaviour thus they need to adjust their weights. I am investigating to what extent this behaviour can constitute a basic mechanism of social teaching: if there are some experienced robots the new ones do not have to learn a lot because the others are compensating for their mistakes. I can make a hypothesis that there is a critical mass of new agents when the new ones need to start to learn as well. The reader must remember that this model is not based on knowledge transfer models of social colonies: the agents are not learning by imitation. Learning by imitation requires rather complex functions which are not present in simple organisms or agents like the one used in this simulation.

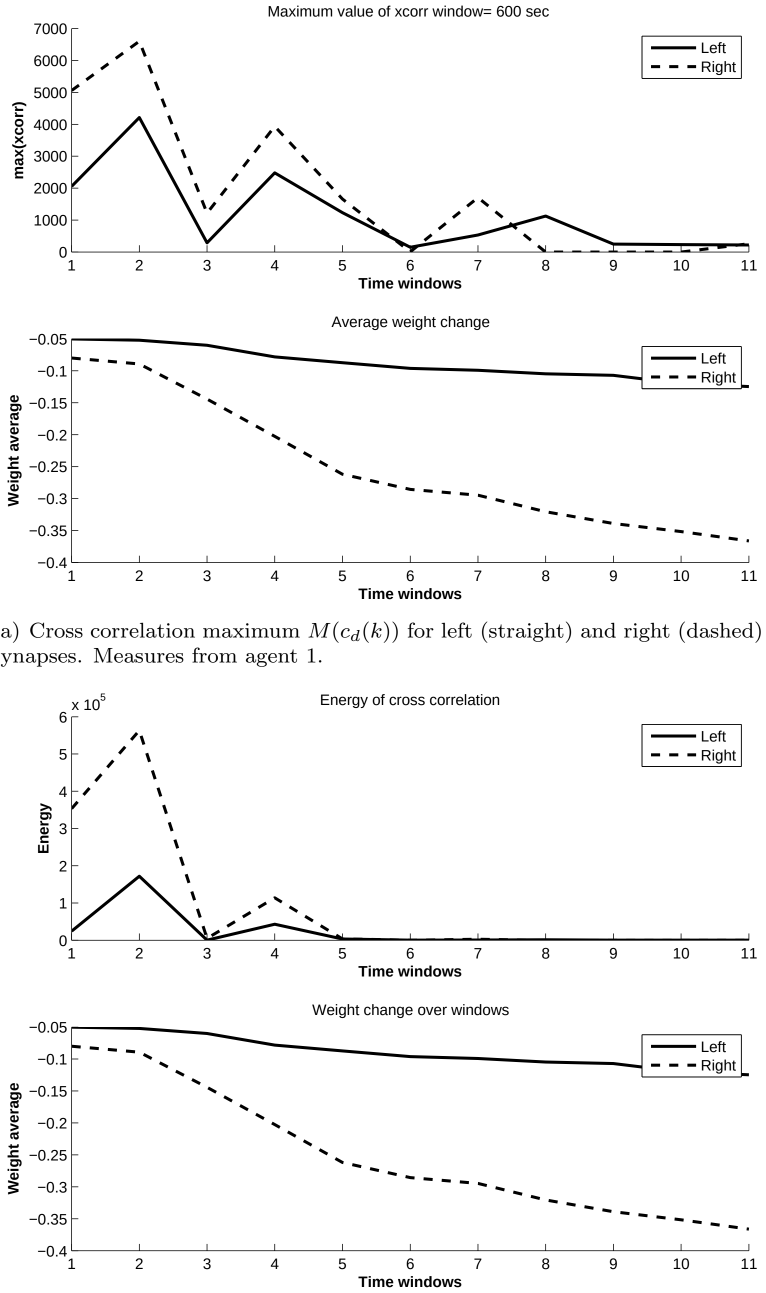
Fig. 3.23 shows the $M(xcorr_{left}(k)), M(xcorr_{right}(k))$ values for an experienced agent. When the new agent is dropped in the arena at time window 6, the agent sees it by a peak in the cross correlation and thus needs to adjust its weights. Unfortunately I did not have any more time to carry extensive analysis, but it would have been interesting how the performance of newly introduced agents is affected by the other experience agents. Probably the newly



(a) Cross correlation maximum $M(c_d(k))$ for left (straight) and right (dashed) synapses. Measures from agent 0.

(b) Energy $E(xcorr_d(k))$ for left (straight) and right (dashed) synapses. Measures from agent 0.

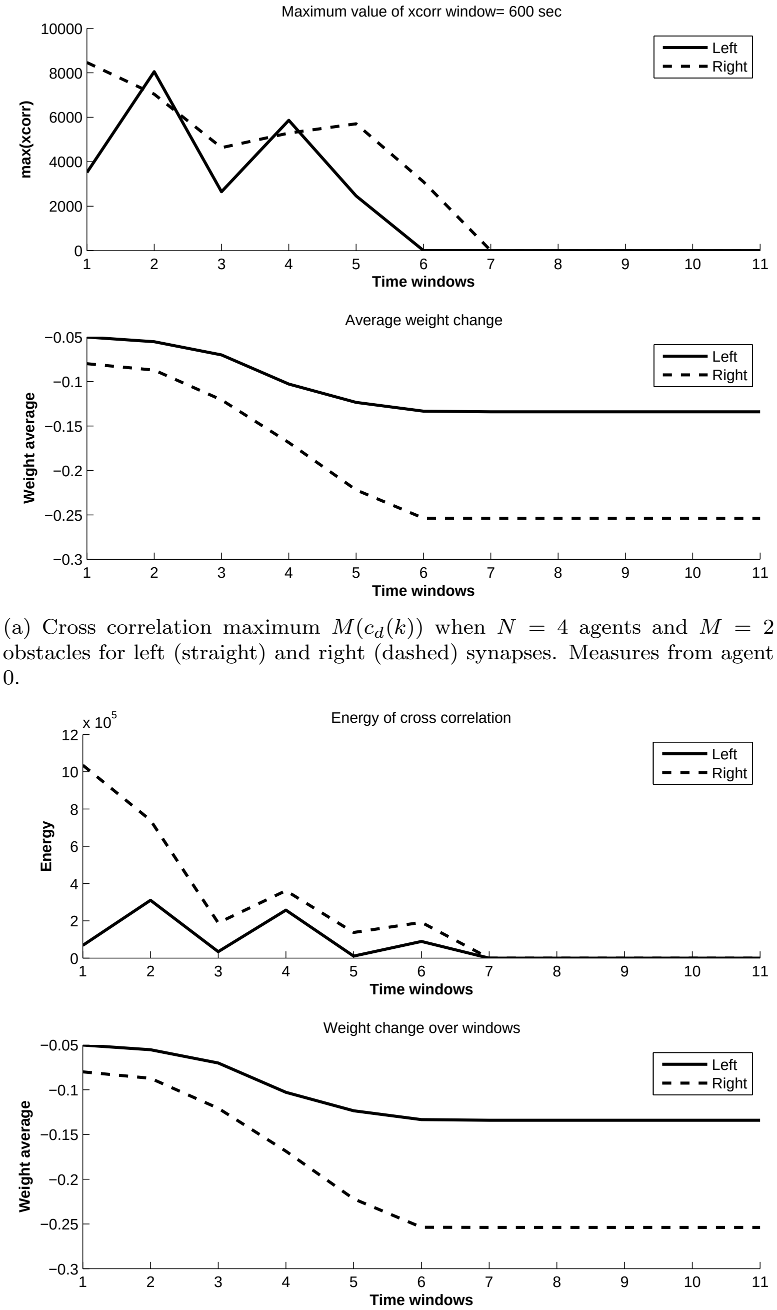
Figure 3.17: Information measure analysis for $N = 2$ agents and $M = 2$ obstacles computed in $k = 1, ..., 11$ time windows. Learning is switched on for all agents when $(k > 1)$.



(a) Cross correlation maximum $M(c_d(k))$ for left (straight) and right (dashed) synapses. Measures from agent 1.

(b) Energy $E(xcorr_d(k))$ for left (straight) and right (dashed) synapses. Measures from the agent 1.

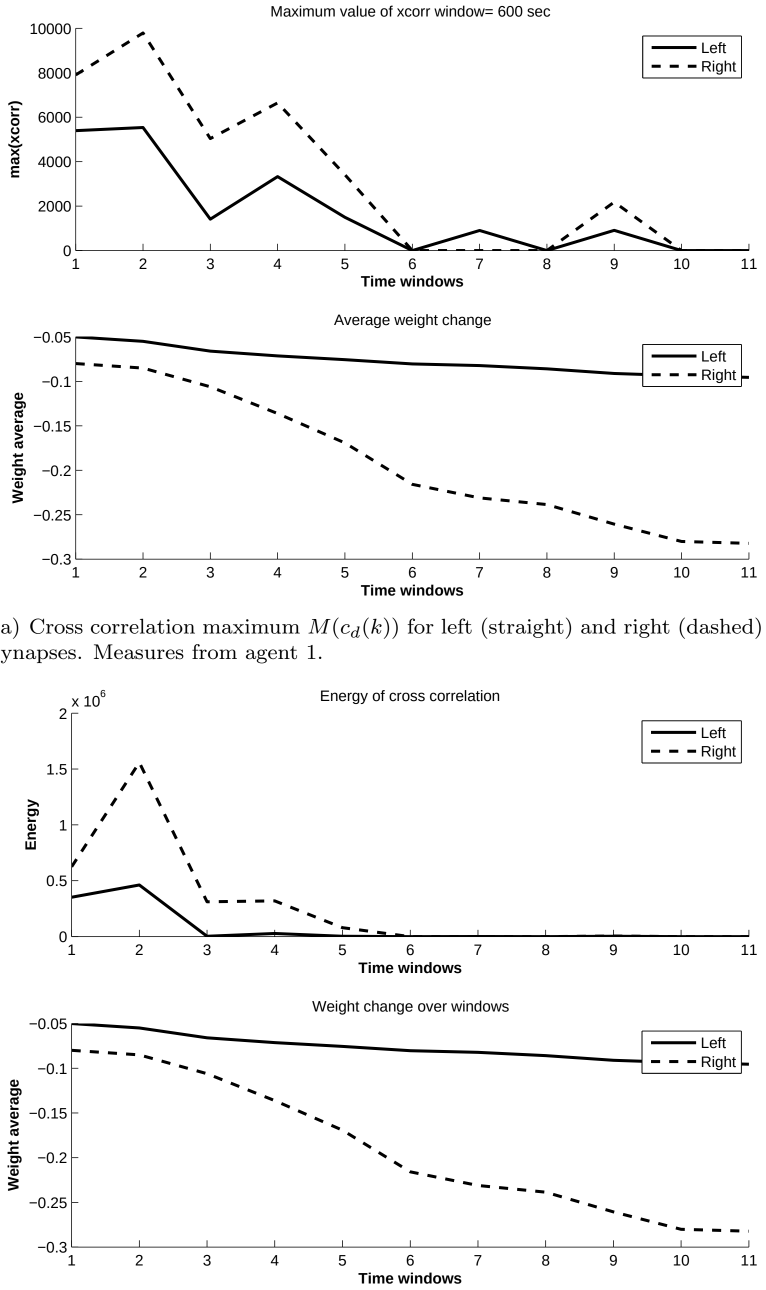
Figure 3.18: Information measure analysis for $N = 2$ agents and $M = 2$ obstacles computed in $k = 1, ..., 11$ time windows. Learning is switched on for all agents when $(k > 1)$.



(a) Cross correlation maximum $M(c_d(k))$ when $N = 4$ agents and $M = 2$ obstacles for left (straight) and right (dashed) synapses. Measures from agent 0.

(b) Energy $E(xcorr_d(k))$ when $N = 4$ agents and $M = 2$ obstacles for left (straight) and right (dashed) synapses. Measures from the agent 0.

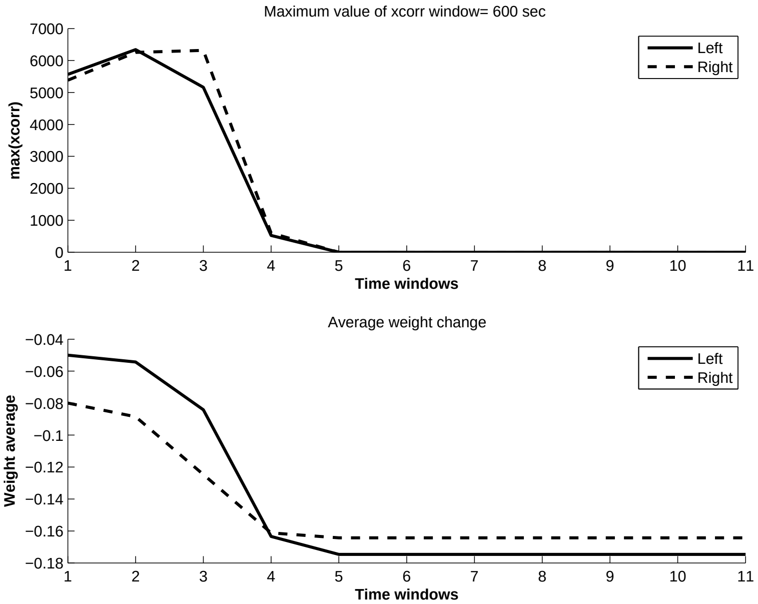
Figure 3.19: Information measure analysis for $N = 4$ agents and $M = 2$ obstacles computed in $k = 1, ..., 11$ time windows. Learning is switched on for all agents when $(k > 1)$.



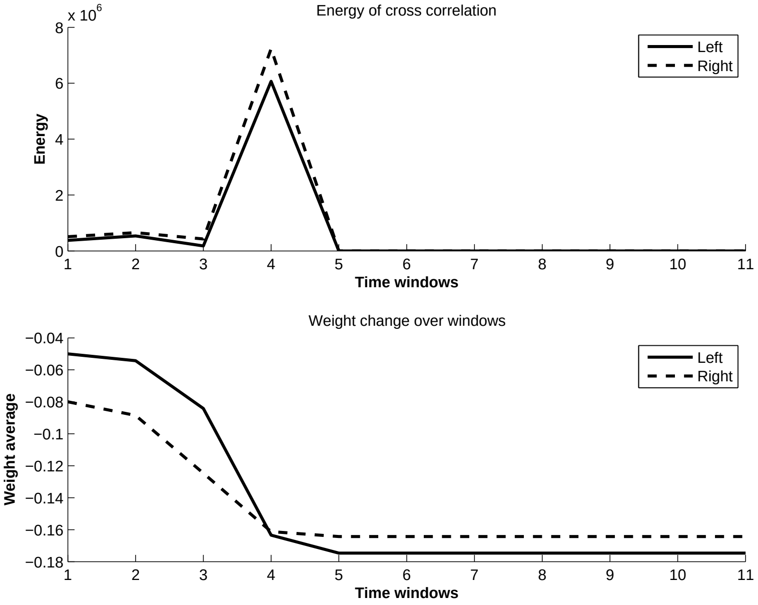
(a) Cross correlation maximum $M(c_d(k))$ for left (straight) and right (dashed) synapses. Measures from agent 1.

(b) Energy $E(xcorr_d(k))$ for left (straight) and right (dashed) synapses. Measures from the agent 1. Learning starts after the first window.

Figure 3.20: Information measure analysis for $N = 4$ agents and $M = 2$ obstacles computed in $k = 1, ..., 11$ time windows. Learning is switched on for all agents when $(k > 1)$.

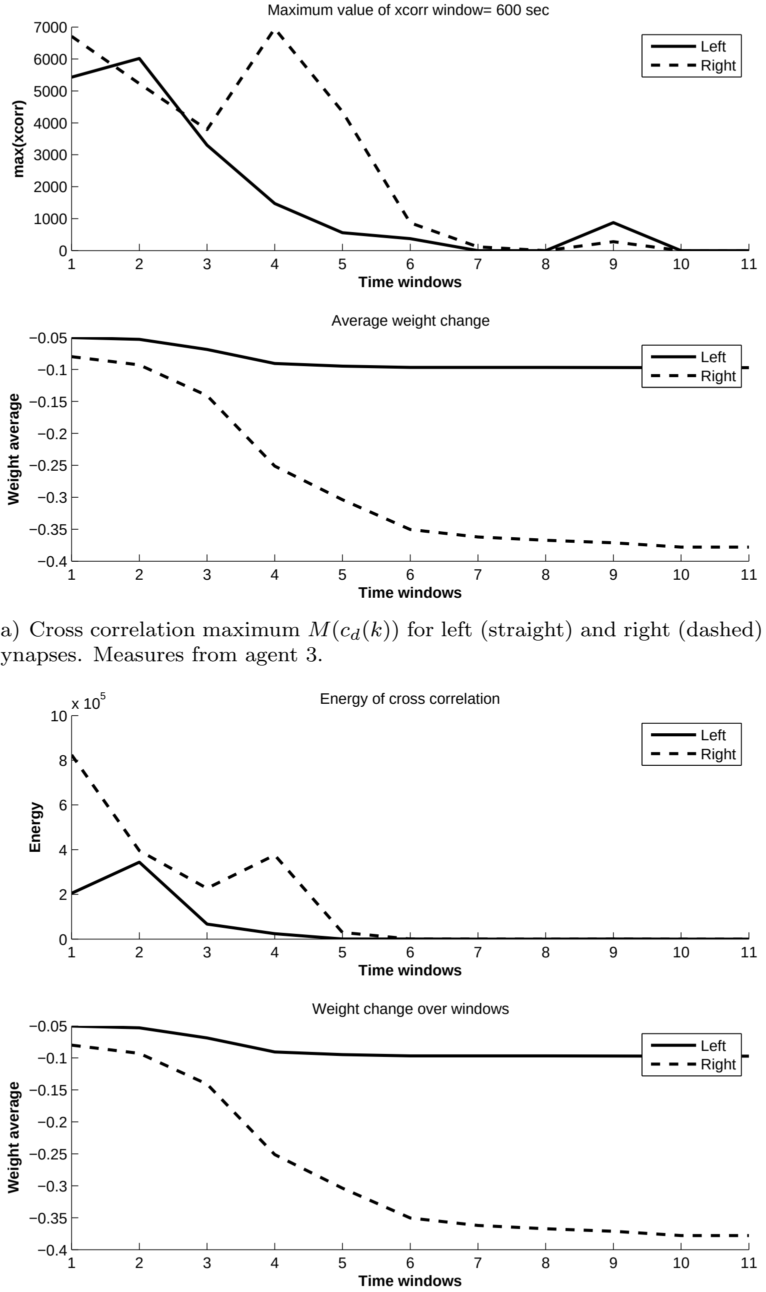


(a) Cross correlation maximum $M(c_d(k))$ for left (straight) and right (dashed) synapses. Measures from agent 2.



(b) Energy $E(xcorr_d(k))$ left (straight) and right (dashed) synapses. Measures from the agent 2.

Figure 3.21: Information measure analysis for $N = 4$ agents and $M = 2$ obstacles computed in $k = 1, ..., 11$ time windows. Learning is switched on for all agents when $(k > 1)$.



(a) Cross correlation maximum $M(c_d(k))$ for left (straight) and right (dashed) synapses. Measures from agent 3.

(b) Energy $E(xcorr_d(k))$ for left (straight) and right (dashed) synapses. Measures from the agent 3.

Figure 3.22: Information measure analysis for $N = 4$ agents and $M = 2$ obstacles computed in $k = 1, ..., 11$ time windows. Learning is switched on for all agents when $(k > 1)$.

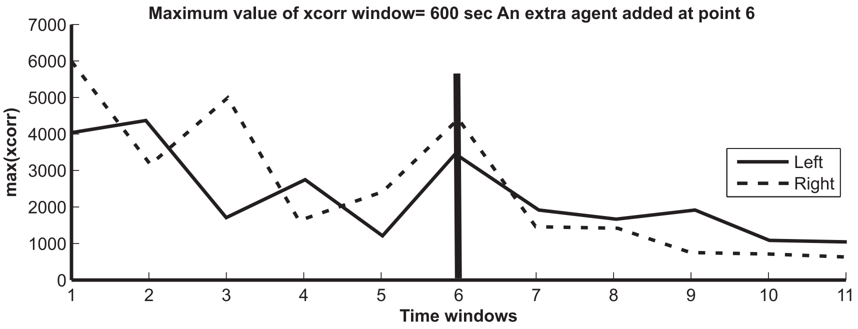


Figure 3.23: A new agent is introduced for t=6, the other agents needs to adjust their weights to the new “unpredictable” friend.

introduced agents wouldn’t require a lot of learning as the others are already successfully avoiding.

3.2.8 Results: simple case results

In this simple case there are 2 agents navigating in a rectangular space with a number of 2 obstacles that are placed randomly in the environment (see Appendix 5.2.6). Every simulation is run for $T = 30$ minutes with a time step of $\Delta = 0.01s$, the window for the correlation is $W = 10$ seconds, therefore $cc(t)$ is computed for $t = 1, 2, ..., 180$.

Anticipatory information is averaged over 100 simulations: every simulation is randomised in terms of the robots’ and obstacles’ initial positions. In the first time window the learning of the agents is switched off and $cc(0) \simeq 0.7071$ because the reflex of the agent is already preventing a full force impact, whereas for $cc(t > 0) \leq 0.7071$ because the impact of the collision can only be equal or decreased.

Fig.3.24(B) shows that AI increases and stabilises to 6 bits when agents have learned successfully: they are using 6 bits in the predictive loop to reduce the reflex. Instead non learning agents are not using any bit to reduce the reflex. The noise visible in Fig.3.24 is due to the random repositioning of the obstacles and random collisions. In a perfect world if agents were able to avoid perfectly $cc(t) \simeq 0$ and so $AI(t) \rightarrow \infty$.

3.2.9 Results: simplified social model results

I apply now the **AI measure** to a social system such as the one published in Di Prodi et al. (2008) and described in section 3.1 where a social system whose task is cooperative food foraging. As for the avoidance case the agents learn how to use the distal sensors to approach food (to increase their energy) or other agents (to get their energy). Agents can forage directly from the food patches (see Fig.3.25(B)) or reduce the energy of other agents who have previously got food (see Fig.3.25(D)) .

Thus every agent has two competitive signals: one from the food patches and one indicating the energy level of the other agents. Indeed when antennas are in contact with another agent

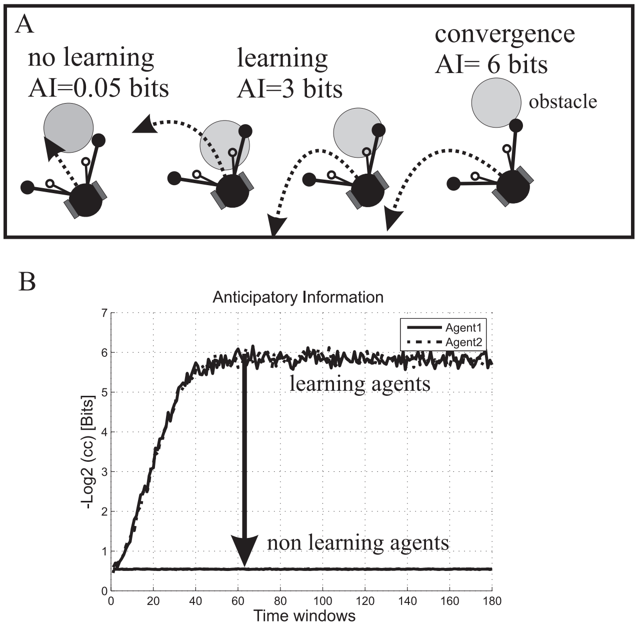


Figure 3.24: **A)** Two agents are learning to avoid obstacles in a closed rectangular world. At the very left the agent is only reacting to the obstacle thus $AI \simeq 0$, at the very right the agent has learnt successfully how to exploit the predictive signal u_1 to avoid the contact thus $AI > 0$. **B)** Anticipatory information computed for 2 learning agents. When learning is stable it reaches a baseline level, $AI(t) \simeq 6$ bits. If agents are not learning the anticipatory information is low $AI(t) \simeq 0.5$ bits, because $cc(0) \simeq 0.7071$.

with high energy they produce impulses on $x_{1,e}$ for far contacts and on $x_{0,e}$ for near contacts, whereas when they are in contact with a food patch they produce impulses on $x_{1,f}$ for far contacts and on $x_{0,f}$ for near contacts.

Therefore the agent has 2 learning weights $\omega_{1,e}$ for energy and $\omega_{1,f}$ for food that both contribute to the motor output. When the simulation starts, all agents have $\omega_{1,e} = \omega_{1,f}$ and therefore they approach any object because according to the situation they will choose the food or the nearby agent. Nevertheless during the simulation some agents (the seekers) will become more attracted by the food $\omega_{1,e} < \omega_{1,f}$, while the others (the parasites) will become more attracted by the other agents with energy $\omega_{1,e} > \omega_{1,f}$. An agent changes class or behaviour when the weights are swapped i.e. a seeker $\omega_{1,e} < \omega_{1,f}$ becomes a parasite $\omega_{1,e} > \omega_{1,f}$ (or vice-versa) thus contributing to the system instability.

The bar diagram of Fig.3.25(E) shows for every time window how many times this swap has happened.The system self-stabilises to a number of seekers and parasites whose number depends on the available resources: in this case with 4 food resources there are 6 parasites and 4 seekers. With more resources e.g. 10, there will be 6 seekers and 4 parasites.

Luhmann theorised that sub-systems are formed to reduce the complexity of the perceived environment: in this case this means that agents are discarding part of the closed loop information. This process is shown in Fig.3.25 by computing AI for the energy and for the food signal. The $AI(u_{1,e}, u_{0,e}) = AI_{energy}$ represents the anticipatory information for the energy attraction and the $AI(u_{1,f}, u_{0,f}) = AI_{food}$ for the food attraction. For seekers the $AI_{food} > AI_{energy}$ increases as the system differentiates and conversely for the parasites $AI_{energy} > AI_{food}$. In term of information it means that seekers are using $AI_{food} - AI_{energy} = 2$ bits more to reduce the energy signal, whereas parasites are using $AI_{energy} - AI_{food} = 2$ bits more to reduce the food.

3.2.10 Results: differentiation and information measure

I then computed the $xcorr(k)$ measure in the same social model (Di Prodi et al., 2008). The Fig.3.26 is the outcome of the computation on a group of $N = 10$ robots and $M = 2$ food places, where the systems develops 6 parasites and 4 seekers. In Fig.3.26 (A) for the 4 seekers the information measure of the food's signal is smaller than the average information measure of the agent signals. This means that the seekers learn to use only one signal out of two in order to simplify the closed loop model. In Fig.3.26 (B) for the 6 parasites the information measure of the agents' signal is smaller than the average information measure of the food's signals. This means that the parasites learn to use only one signal out of two in order to simplify the closed loop model. As I expected when the system is not stable Fig.3.26 (C) -the agents are switching their class- the information measure for the food signals and the agent signals are crossing each other, but when the system becomes stable - switching rate is approaching 0- I can see that the two information measures start to separate.

The results shows how each agent selects only one input out of two and thus makes its environment more predictable. Luhmann theorised that in society, specialised groups emerge when they reduce the uncertainty of the environment-system boundary in a recursive process. The model's results validate his theory because agents are selecting to use one information rather than the other one to simplify the model of the environment's loop.

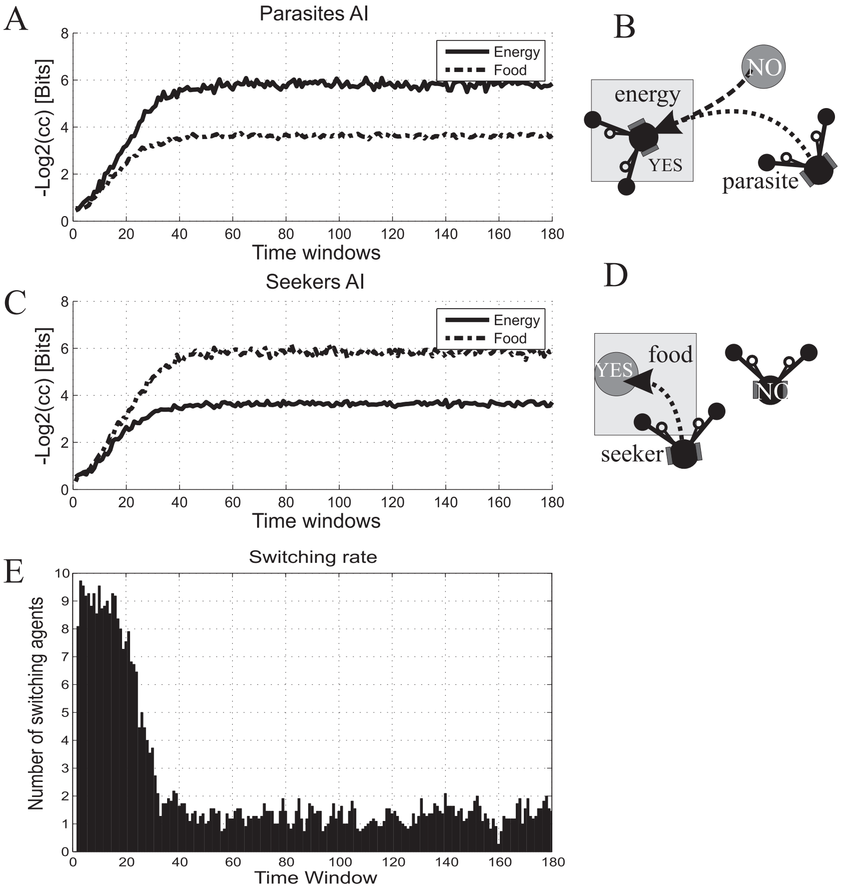


Figure 3.25: B) A parasite is an agent that prefers an energy signal to a food one. D) A seeker is an agent that prefers a food signal to an energy one produced by another agent. B) For parasites the anticipatory information for food is less than that one of the energy. C) For seekers the anticipatory information for energy is less than that one of the food. E) There are a total of $N = 10$ agents of whom 6 turn into parasites, and 4 turn into seekers after 20 time steps. The system stabilisation is a function of time: at the very beginning agents are using both signals and their behaviour is unpredictable because they are switching between the 2 competitive behaviours. After 20 time windows the agents have a more predictable behaviour resulting by the selection of the information. Minor oscillations are again due to the noise resulting from the interaction between learning agents.

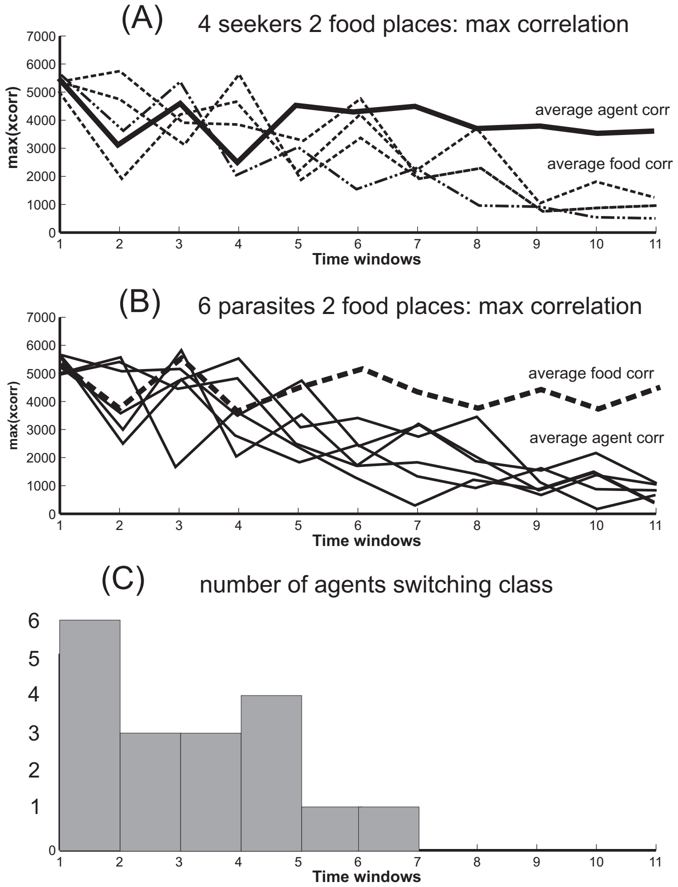


Figure 3.26: The information measure computed on a group of $N = 10$ robots. (A) information measure computed on the 4 seekers and compared to the average measure of the 6 parasites. (B) information measure computed on the 6 parasites and compared to the average measure of the 4 seekers. (C) The switching rate: how many agents changes their class for every time window. The system is stable when the switching rate is 0, in this case after 7 windows.

3.2.11 Discussion

Devising a measure which affects learning in agents in a closed loop is a challenging task. Therefore it is necessary to make the right assumptions: in my case the agent is performing input control (as described in Section 2.2.2) and so is necessary to measure how the predictor changes in respect of the reflex. The same **maxcorr** or **AI measure** cannot be applied to agents which performs output control such as in the traditional artificial neural networks trained on the desired output and not input. Shannon's entropy was also applied to closed loop controllers as in Touchette and Lloyd (2000b) which considers the perception action-loop in terms of a communication channel-like model. Also Polani et al. (2004) recently has been using the same approach: the perception-action loop enables an agent to use its actuators as a channel to transmit information into the environment. The information can later be acquired from the environment by the same agent or other agents. In our measure there is no such transfer channel: temporal information relevant to the agent is computed only at the inputs which contains the feedback of the outputs. It is simple and easy to compute and it does not require the modelling of the agent's controller as an information channel. Recently, Polani et al. (2001) has been working on evolving sensors and has introduced an operational notion of Shannon-type quantification of relevant information:

- it quantifies relevance with respect to a given agent or decision system.
- it yields a measure for the usefulness of sensors or about an agent's state of knowledge with respect to a POMDP (Partial Observable Markov Decision Process)

The problem of this approach is that it requires a discrete formulation of an agent, and its application to a general controller is not possible. The limit of the **maxcorr** measure is that it is dependent on the particular implementation of our controller and cannot be applied to other adaptive algorithms unless they are based on a similar temporal input structure. The second issue of the **maxcorr** is that - although normalised - it does not give an indication of the capabilities of the agent. The **AI measure** was then introduced to quantify the learning behaviour in terms of bits but it does not provide an axiomatic framework and thus is mathematically weak. Therefore in Section 3.3, I developed a more advanced measure based on the concept of information flow which is inspired by the work of Pfeifer et al. (2008).

3.3 Information flow for adaptive controllers

This Section describes an improvement of the previous information measure described in Section 3.2. The new theoretical framework, based on Shannon's communication theory and on Ashby's law of requisite variety, is suitable for artificial agents using predictive learning. The framework quantifies the performance constraints of a predictive adaptive controller as a function of its learning stage. In addition, I formulate a practical measure, based on information flow, that can be applied to adaptive controllers which use Hebbian learning, input correlation learning (ICO/ISO) and temporal difference learning. The framework is also useful in quantifying the social division of tasks in a social group of honest, cooperative food foraging, communicating agents. Simulations are in accordance with Luhmann, who suggested that adaptive agents self-organise by reducing the amount of sensory information or, equivalently, reducing the complexity of the perceived environment from the agents perspective.

3.3.1 Introduction: Ashby's theory

Information measures are usually defined for input/output systems where they determine the quality of the transmission. Behaving agents, however, act as closed loop systems in which there is no clearly defined difference between input and output. What matters most for the organism is to compensate for disturbances introduced by the environment in the perception action loop. If there is no disturbance, the organisms cannot differentiate between themselves and the environment. Consequently, the concept of information in these systems needs to be revised (Porr and Wörgötter, 2005).

A method for defining closed loop information has been proposed by Ashby (1956) as the so called *requisite variety* . The measure is based on the premise that closed loop systems aim to maintain a desired state. The goal of a feedback loop is then to minimise the deviation from the desired state i.e. the number of bits required to successfully compensate a disturbance acting on the forward loop. In this way, the method quantifies the variety, or bits, originating from the disturbance. For example, if the disturbance has a variety of 10 bits and survival requires a desired state of 2 bits, then the reaction to that disturbance must provide a variety of 8 bits. Ashby then proved that error controlled closed loop systems (like PID controllers discovered by Stuart 1984) cannot achieve perfect regulation. More recently, Touchette and Lloyd 2000b in Theorem 10 proved that the entropy reduction achieved by a closed loop system is bounded by the entropy reduction achieved by the open loop control plus the mutual information gathered by the estimation of the state. However the advent of predictive controllers, such as Q-learning (Sutton and Barto, 1998), that predict future states, requires an extension of the information theory for predictive learning.

In this section, I present an extension to the law of requisite variety, called *the predictive requisite variety*, that quantifies the theoretical limits of control (as well as providing a performance index) for predictive adaptive controllers. I argue that a predictive adaptive controller acts as a reactive system before learning and as an open feed-forward system after learning. A reactive system comprises an error controlled closed loop and is non optimal because it only reacts after a deviation from its desired state has happened. The environment usually contains predictive signals which can help the agent to react before the error is presented (Verschure et al., 2003). Thus, biologically inspired controllers can be provided with a predictive signal (like vision) and a reflexive signal (like touch). Learning then has the task of avoiding the trigger of the reflexive reaction - thus creating an open loop forward controller which discards the information of the reflexive signal.

Learning is then quantified by the increase in the information flow of the predictive loop and by a corresponding decrease in the information flow of the closed loop. Information flow, or transfer entropy, is not a new idea (see for example (Polani, 2010; Schreiber, 2000)) but it has never been applied to predictive agents in order to assess their learning performance. The analysis of a predictive agent with a single behaviour, say for example obstacle avoidance, can be done by calculating the information flow of the sensory-motor loop.

Analysis becomes more complicated when an agent is provided with a set of competitive behaviours in a social scenario where agents use predictive learning- see, for example, ISO (Porr and Wörgötter, 2003) or ICO (Porr and Wörgötter, 2006) - and are therefore learning from each other. The task of the social system in this analysis is cooperative food foraging in which every agent has 3 adaptive behaviours which are: avoidance for obstacles, attraction to food disks and attraction to others with food. Agents communicate honestly, always signalling to others when they find food. When the social system is adapting, it self-organises into 2 sub-systems each described by a dominant behaviour: seekers have a dominant attraction for food disks, parasites have a dominant attraction to others with food. The information flow explains how the social system divides itself into sub-systems by looking at the information processing of every agent. Luhmann (1995) proposed that differentiation of social systems is caused by a decrease in information processing of each subsystem and this is consistent with my information flow measurements.

The following sections covers the topics: regulation and entropy (as defined originally by Ashby), a new information measure for predictive learning, a simulation model with social adaptive agents, results and a discussion.

3.3.2 Methods: Ashby's law of requisite variety

First, it is necessary to review Ashby's Law of Requisite Variety for the forward (see Fig.3.27(B)) and closed loop controller (see Fig.3.27(A)). Fig.3.27 uses the same notation introduced by Ashby:

- **D** is the finite state machine whose states are the disturbances from the environment
- **E** is the finite state machine whose states are the essential variables partitioned in $E = \eta \cup \overline{\eta}$, where η is a partition of desired states or goals of the organism and its complementary partition $\overline{\eta}$ represents the non-desired states.
- **R** is the finite state machine whose states are the available regulations/actions that the organism can perform
- **T** is the finite state machine whose states are the set of possible states of the environment

In this work I consider deterministic finite state machines but the analysis can also be extended to Markov processes as in Booth (1967). It is very important for my analysis to understand that only the forward controller can achieve perfect regulation whereas the closed loop controller cannot because the reflex always comes too late. Ashby (1956) stated that a good controller R blocks the flow of variety[2] from disturbances D to essential variables E: if R is a regulator, the insertion of R between D and E decreases the variety that is transmitted from D to E. An organism can be described by a body R with goals to be achieved η and an environment T which forms a closed loop between actions and sensors. As an analogy, the organism is a perfect regulator if is able to keep the essential variables E within a desired sub-set η in spite of the disturbances D -thus having a null entropy for E, $H(E) = 0$.

Definition and properties If no regulator R is provided (see Fig.3.27(C)), the disturbance D tends to drive E_0 outside a set of desired states η by means of the environment T. Thus, in the worse case, the disturbance completely controls the status of the organism:

$$H(D) = H(E_0) \tag{3.75}$$

which means all the disturbances is transferred intact to the organism. The regulator R can be connected in a feed-forward configuration as in Fig.3.27(B) or in a closed loop configuration as

[2] Ashby defines variety precisely as the number of different states a variable can take and is equivalent to Shannon's entropy H measured in bits.

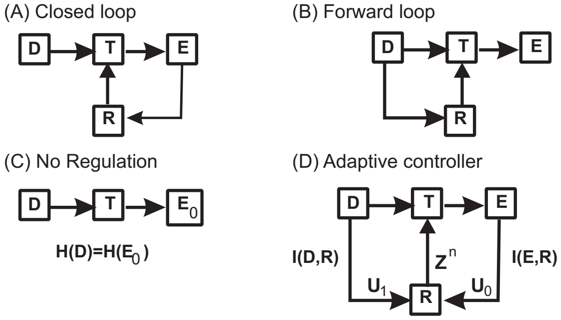


Figure 3.27: (A) The organism with a closed loop controller. (B) The same organism with an forward controller.(C) The organism before regulation. (D) An adaptive controller is a mix of forward and closed loop control. Every block is a finite state machine whose inputs are indicated by incoming arrows and outputs are indicated by outgoing arrows.

in Fig.3.27(A). The performance of the forward regulator is measured by the maximum entropy reduction $\Delta H^{max}_{forward}$ which is the difference between the entropy of the essential variable $H(E_0)$ before regulation and after regulation $H(E)$.

$$\Delta H^{max}_{forward} = H(E_0) - minH(E) \tag{3.76}$$

The maximum entropy reduction in the forward condition $\Delta H^{max}_{forward}$ can be calculated by using the Law of Requisite Variety:

$$H(E) \geq H(D) + H(R|D) - H(R) \tag{3.77}$$

where $H(R|D)$ is the regulator noise[3]. Thus:

$$\Delta H^{max}_{forward} = H(R) - H(R|D) \tag{3.78}$$

because combining Eq.3.76 and Eq.3.77 gives:

$$\Delta H^{max}_{forward} = H(E_0) - H(D) - H(R|D) + H(R) \tag{3.79}$$

Considering the initial condition in Eq.3.75 I obtain Eq.3.78:

$$\Delta H^{max}_{forward} = H(D) - H(D) - H(R|D) + H(R) = H(R) - H(R|D) \tag{3.80}$$

The quantity $\Delta H^{max}_{forward}$ in Eq.3.78 tells us that better performance can be achieved by either increasing the regulation entropy $H(R)$ or by decreasing the controller noise $H(R|D)$. A closed loop controller cannot achieve perfect regulation ($H(E) = 0$) as it requires a deviation from the desired state η to work $H(E) > 0$. Thus, the disturbance transmits all its entropy to the essential variable $H(D) = H(E)$ and no entropy reduction can be achieved:

$$\Delta H^{max}_{close} = 0 \tag{3.81}$$

[3] If the controller is not noisy $H(R|D) = 0$

When $H(E) = 0$, R blocks the information flow in the channel $D \to E$ and thus no information is transmitted to R for the regulation task: the regulator R is asserting a perfect control on E without knowing the status. This property was proved by contradiction in Ashby (1956). In the next section I extend the law of requisite variety for adaptive controllers.

3.3.3 Methods: the law of adaptive requisite variety

An adaptive controller (see Fig.3.27(D)) is a mix of a forward (Wang et al., 2005) and closed loop controllers (Stuart, 1984) because R now has 2 inputs: D and E. I can think of D as a predictor of the deviation of E, because D transfers its entropy to E by means of the environment T.

In order to explain the new law, I introduce the mutual information $I(E, R)$ for the closed loop channel $E \to R$ with the corresponding channel capacity $C_{E,R}$:

$$I(E,R) = H(E) + H(R) - H(E,R) \tag{3.82}$$
$$C_{E,R} = \max_{p(E)} I(E,R) \tag{3.83}$$

the mutual information $I(D, R)$ for the forward channel $D \to R$ with the corresponding channel capacity $C_{D,R}$:

$$I(D,R) = H(D) + H(R) - H(D,R) \tag{3.84}$$
$$C_{D,R} = \max_{p(D)} I(D,R) \tag{3.85}$$

The channel capacity of the regulator channel $D \to T$ is then $C_{R,T}$.

The adaptive controller (denoted ada) begins as a closed loop controller with $\Delta H_{ada}^{max}(before) = H_{close}^{max}$ (see Eq.3.81) as it mainly uses the $E \to R$ reflex channel and blocks the $D \to R$ predictor channel whose mutual information is very low. In summary:

$$0 < I(E,R) \leq C_{E,R} \tag{3.86}$$
$$I(D,R) \simeq 0 \tag{3.87}$$
$$\Delta H_{ada}^{max}(before) = 0 \tag{3.88}$$

The adaptive controller achieves perfect regulation (see Eq.3.78) when

$$\Delta H_{ada}^{max}(after) = H_{forward}^{max} \tag{3.89}$$

because it blocks the $E \to R$ reflex channel and opens the $D \to R$ predictor channel. To summarise:

$$0 < I(D,R) \leq C_{D,R} \tag{3.90}$$
$$I(E,R) \simeq 0 \tag{3.91}$$
$$\Delta H_{ada}^{max}(after) = H(R) - H(R|D) \tag{3.92}$$

If I assume realistically that the regulator has a common channel capacity $C_{E,R} = C_{D,R} = C_{R,T}$, the constraint for learning becomes:

$$I(E,R) + I(D,R) \leq C_{R,T} \tag{3.93}$$

thus an adaptive controller can achieve optimal regulation $\Delta H^{max}_{ada}(after)$ when it is able to compensate the mutual information of the closed loop $I(E, R)$ with the mutual information of the forward controller $I(D, R)$. An imperfect regulator will likely work in the sub-optimal regime $I(D, R) < I(E, R)$. So to quantify the performance of an adaptive predictive controller I have to compute the mutual information $I(D, R)$ and $I(E, R)$. This is however not always possible because it is hard to identify the reflex channel and the predictor channel. Therefore in the next section I use an approximation of these 2 quantities using the concept of information flow.

3.3.4 Methods: information flow for adaptive predictive controllers

Looking at Fig.3.27(D), I can estimate $I(E, R)$ by computing the information flow of the reflex-output channel $Z^n \to U_0$ and $I(D, R)$ by computing the information flow of the predictive-output channel $Z^n \to U_1$. I denoted them as:

$$MI^n_{U0} = I(Z^n, U_0) \leftrightarrow I(E, R) \tag{3.94}$$

$$MI^n_{U1} = I(Z^n, U_1) \leftrightarrow I(D, R) \tag{3.95}$$

where $U0$ is the reflex input, $U1$ is the predictor input and Z^n the extended output:

$$Z^n = [z(k)z(k+1)\dots z(k+n-1)] \tag{3.96}$$

which contains n outputs of the agent and U the random variable describing the temporal signal $u(k+n)$ which is the input of the agent resulting from n previous actions at time k as described in Polani et al. (2004); Pegors et al. (2005). The double arrows indicate the correspondence between the mutual information computed and the diagram in Fig.3.27(D).

Fig.3.28(A) shows an organism composed of 3 ICO (Porr and Wörgötter, 2006) controllers and the corresponding information flow measures for every controller. Each ICO controller takes 2 continuous inputs $U0, U1$ and one continuous output Z_n.

ICO correlates the predictive signal u_1[4] with the derivative of the reflexive signal u_0 according to the formula:

$$\frac{d\omega_1}{dt} = \mu \cdot u_1 \cdot \frac{du_0}{dt} \tag{3.97}$$

where ω_1 is the gain of the predictive signal u_1 and μ is the learning speed (see Fig.3.28(C)).

Since the ICO controller works in continuous mode, the input and output signals must be discretised in order to compute the information flow and channel capacity (see Simulation Details). The two measures MI^n_{U0}, MI^n_{U1} are used to compute the channel capacities $C_{E,R}$ and $C_{D,R}$:

$$\zeta^n(Z^n \to U0) = \max_{p(Z^n)} MI^n_{U0} \leftrightarrow C_{E,R} \tag{3.98}$$

$$\zeta^n(Z^n \to U1) = \max_{p(Z^n)} MI^n_{U1} \leftrightarrow C_{D,R} \tag{3.99}$$

In the simulations in the next section, I will estimate the mentioned quantities for individual agents of a social group.

[4] u_1 and u_0 indicates temporal signals $u_1(t)$ and $u_0(t)$

3.3.5 Methods: information flow applied to MISO controller

The previous measures are applied to a social system where all agents learn continuously from each other and from the environment. This scenario is very interesting because the social system is able to self-organise by forming 2 sub-systems with task division. The social system described in Di Prodi et al. (2008) is composed of N identical agents and M food disks randomly placed in a square world for every simulation (for more details see Appendix 5.2.8). Food disks contain a certain amount of food that is depleted when an agent finds it. The task is cooperative food foraging. The simulated agent is shown in Fig.3.28(B) and has also been used by Kolodziejski and Kulvicius (2009): it is a Braitenberg (Braitenberg, 1984) vehicle with 2 lateral wheels and 2 antennas. By default the agent drives straight forward, with speed $v = 1$ units per time step. It has 2 sensor-pairs, near contact antennas and far contact antennas.

Every agent has a MISO (multiple inputs single output) controller and a variable of 1 bit for the food status. The agent has 3 competitive tasks: avoid obstacles (empty food disks and other agents without food), find food from the disks, find foods from other agents with food. The MISO is composed of 3 parallel ICO controllers (see Fig.3.28(A)) which are provided with a reflex input error u_0, a predictive signal error u_1, a learnt weight ω_1 and an output z. The outputs of the 3 ICO controllers are summed to $z = z_{Av} + z_{Fo} + z_{Af}$ [5] which gives the steering angle: $z = 0$ the robot goes straight forward at speed v, $z > 0$ the robot rotates clockwise, $z < 0$ the robot rotates anti-clockwise. Every simulation is run for $0 \leq k \leq 6 \cdot 10^5$ time steps and is divided in 3 stages. At every stage, each agent produces 6 input time series and 1 output time series $z(k)$ which means that I can calculate the information flow for every pair of reflex-output and predictor-output: MI^n_{U0},MI^n_{U1}. For a single simulation:

1. for $0 \leq k_1 \leq 2 \cdot 10^5$ all agents are reactive ($\mu = 0$). For each agent $i = 1, ..., N$, I have 3 pairs of information flow:
   (a) avoidance: $MI^n_{Av,U1}, MI^n_{Av,U0}$
   (b) food attraction: $MI^n_{Fo,U1}, MI^n_{Fo,U0}$
   (c) others attraction: $MI^n_{Af,U1}, MI^n_{Af,U0}$
2. for $2 \cdot 10^5 < k \leq 4 \cdot 10^5$: every agent is learning $\mu = 1e - 9$ and the weight for every ICO controller $\omega_{1,Av}$,$\omega_{1,Fo}$, $\omega_{1,Af}$ is increasing.
3. for $4 \cdot 10^5 < k_3 \leq 6 \cdot 10^5$: every agent stops learning $\mu = 0.0$ and is using the last weight set at $k = 4 \cdot 10^5$. For each agent I compute again the 3 pairs of the MI^n.

The channel capacities for every agent are computed by providing each isolated output $z = z_{Av}$,$z = z_{Fo}$,$z = z_{Af}$ with a source of independent randomness during a simulation of $2 \cdot 10^5$ time steps for every case. Then I apply the Blahut-Arimoto algorithm (Arimoto, 1972; Blahut, 1972) with a bound error of $\varepsilon = 10^{-11}$ and 5000 maximum iterations to estimate the channel capacity for every agent in the reflex-output loop $\zeta^n(Z^n_k \to U0)$. There is no difference between $\zeta^n(Z^n_k \to U0)$ of every agent so I define ζ^n_{all}. To compute the capacity for the predictor-output loop $\zeta^n(Z^n_k \to U1)$, I use the same approach but preset the weights of every agent to an arbitrary high value to simulate perfect learning:

$$\omega_{1,Av} = 10.0, \omega_{1,Fo} = 10.0, \omega_{1,Af} = 10.0 \tag{3.100}$$

[5] Av stands for obstacle avoidance, Fo for food attraction and Af for attraction to others with food

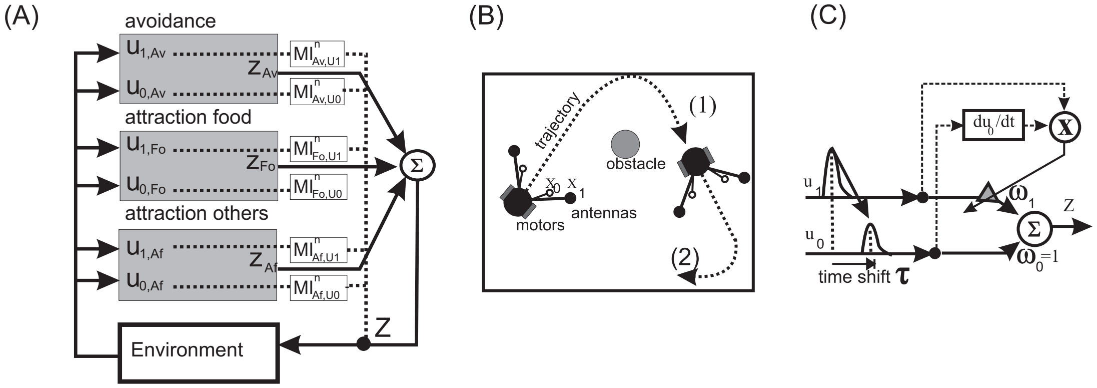


Figure 3.28: (A) MISO controller composed of 3 stacked ICO controllers for avoidance, food attraction and attraction to others. The output of every controller is summed to z. For every controller/behaviour the pair of mutual information is computed between the output and the input MI^n_{U0}, MI^n_{U1}. (B) Agent with short antennas (reflexive inputs, x_0) and long antennas (predictive inputs, x_1). The agent is learning to avoid obstacles. The motor reaction will reduce the intensity of the painful reflex x_0 as well as delay its occurrence. (C) Schematic diagram of the input correlation learning rule and the signal structure (Porr and Wörgötter, 2006). The u_0 and u_1 are, respectively, the difference between the filtered values of the left and right antennas of the agent. During learning the u_0 peak will be shifted in time and reduced in amplitude as the agent learns successfully by increasing the predictor gain ω_1.

and I obtain the same results

$$\zeta^n(Z_k^n \to U1) = \zeta^n(Z_k^n \to U0) = 2.0 \tag{3.101}$$

for $n \geq 2$ as anticipated in Eq.3.98,3.99.

3.3.6 Results

The results of this sections are based on a set of 100 simulations with $N = 10$ agents and $M = 5$ food disks. All agents start with the same weights for every ICO controller $\omega_{1,Av} = 0.1$, $\omega_{1,Fo} = 0.1, \omega_{1,Af} = 0.1$. In stage 3 there are 5 agents with $\omega_{1,Af} < \omega_{1,Fo}$ and 5 agents with $\omega_{1,Af} > \omega_{1,Fo}$. The first group of agents - identified by the indexes 1,7,3,9,2 - is characterized by a strong attractive behaviour for the food disks (see Fig.3.29 (B)), whereas the second group - identified by the indexes 5,8,4,10,6 - is characterized by a strong attractive behaviour for others agent with food (see Fig.3.29 (E)).

I estimate the MI^4 in stage 1 and stage 3 for every agent by using the corrected standard deviation formula (Roulston, 1999). Before learning (Fig.3.29 (A),(D)) the reflex-output loop predominates over the predictor-output loop for both the food attraction behaviour and the others attraction behaviour:

$$MI^4_{Af,U1} < MI^4_{Af,U0} \simeq 0.0025 \tag{3.102}$$

$$MI^4_{Fo,U1} < MI^4_{Fo,U0} \simeq 0.001. \tag{3.103}$$

After learning (stage 3). the configuration is reverted and the predictor-output loop dominates the reflex-output loop for both behaviours as in Fig.3.29(B), (E):

$$MI^4_{Af,U0} \ll MI^4_{Af,U1} \tag{3.104}$$

$$MI^4_{Fo,U0} \ll MI^4_{Fo,U1} \tag{3.105}$$

This result matches my expectations in terms of the increase of $I(D,R)$ and decrease of $I(E,R)$. If I compare the $MI^4_{Af,U1}$ in Fig.3.29(B) to $MI^4_{Fo,U1}$ in Fig.3.29(E) I can see that the agents with indices 1,2,3,4,5 (parasites) have a larger weight $\Delta W_{Af} \simeq 2.0$ (see Fig.3.29(C)) for the attraction to others and, therefore, a larger information flow $MI^4_{Af,U1} > MI^4_{Fo,U1}$, whereas agents with indices 6,7,8,9,10 (seekers) have a larger weight change $\Delta W_{Fo} \simeq 2.0$ for the food attraction and so a bigger $MI^4_{Fo,U1} > MI^4_{Af,U1}$.

Thus, the information measure is directly correlated with the weight change and can be used to quantify the learning performance of a single agent before and after learning. However, it can also be used to quantify the dominant behaviour and, consequently, the self-organising properties of social systems.

In Fig.3.30, I measure the efficiency of the reflex-output and predictive-output loop $MI^4_{Av,U1}$,$MI^4_{Av,U0}$ for the avoidance behaviour in relation to the capacity for the agents $\zeta^4_{all} = 2.0$. Fig.3.30(A) shows that before learning $MI^4_{Av,U0}$ is using 0.25% of the full channel capacity and Fig.3.30(B) shows that after learning $MI^4_{Av,U1}$ is using about 0.45% of the channel capacity. The MI of order $n = 1, 2, 3$ does not provide enough discrimination for the previous analysis because the output history of the agent is too short to be correlated with the inputs. The capacity ζ^n_{all} takes its maximum of 2 bits when $n \geq 2$.

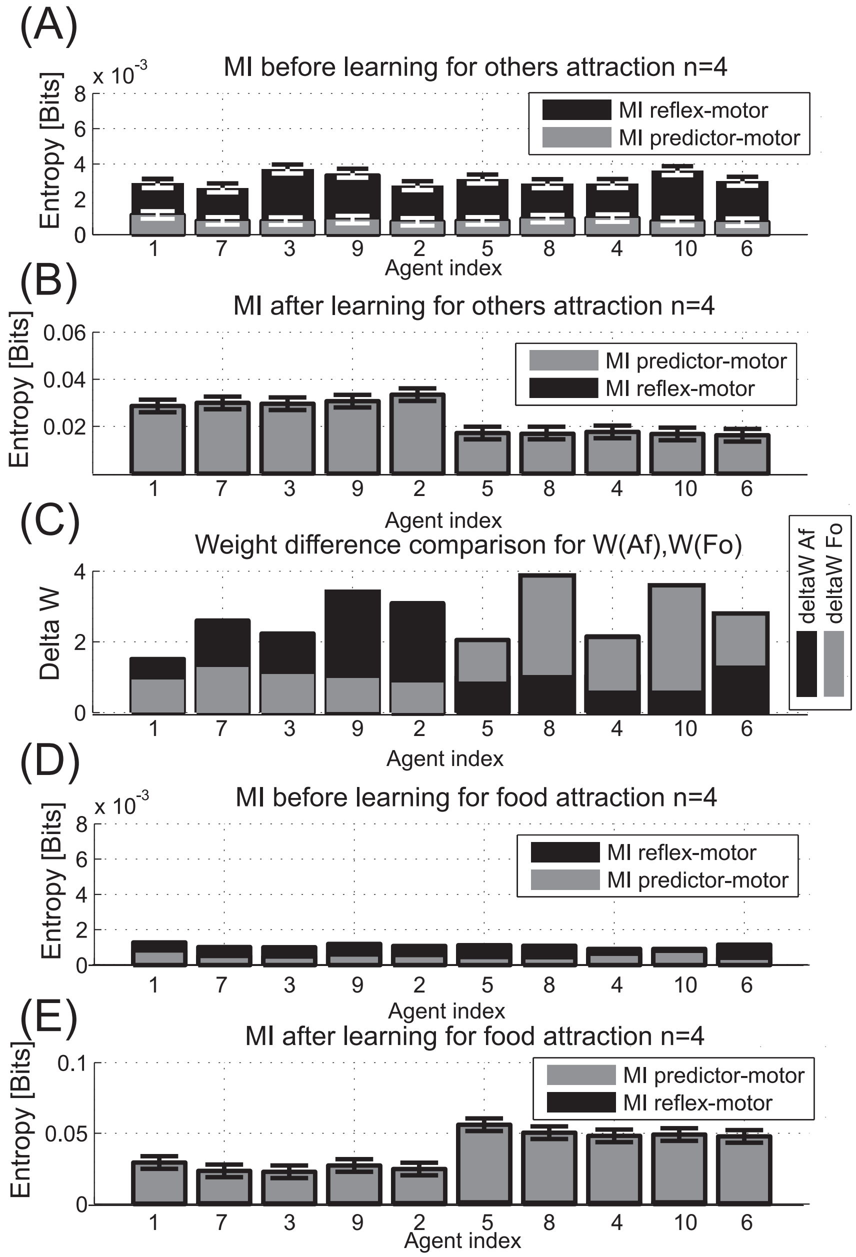


Figure 3.29: **(A)** Information flow before learning for attraction to others $MI^4_{Af,U1}$ (grey bars), $MI^4_{Af,U0}$ (black bars) expressed in bits. **(B)** Information flow after learning for attraction to others in bits. **(C)** Weight difference for every agent: $\Delta W_{Af} = \omega_{1,Af} - 0.1$, $\Delta W_{Fo} = \omega_{1,Fo} - 0.1$ **(D)** Information flow before learning for attraction for food $MI^4_{Fo,U1}$ (grey bars), $MI^4_{Fo,U0}$ (black bars) in bits. **(E)** Information flow after learning for attraction for food in bits. In this typical run from a group of 100 independent simulations, the error bars for each agent indicates the range interval of the measure over the 100 simulations. In this group of simulations the parameters were $N = 10$ and $M = 5$ as a result 5 agents become seekers and 5 agents become parasites. The x-axis contains the number identifying the agent.

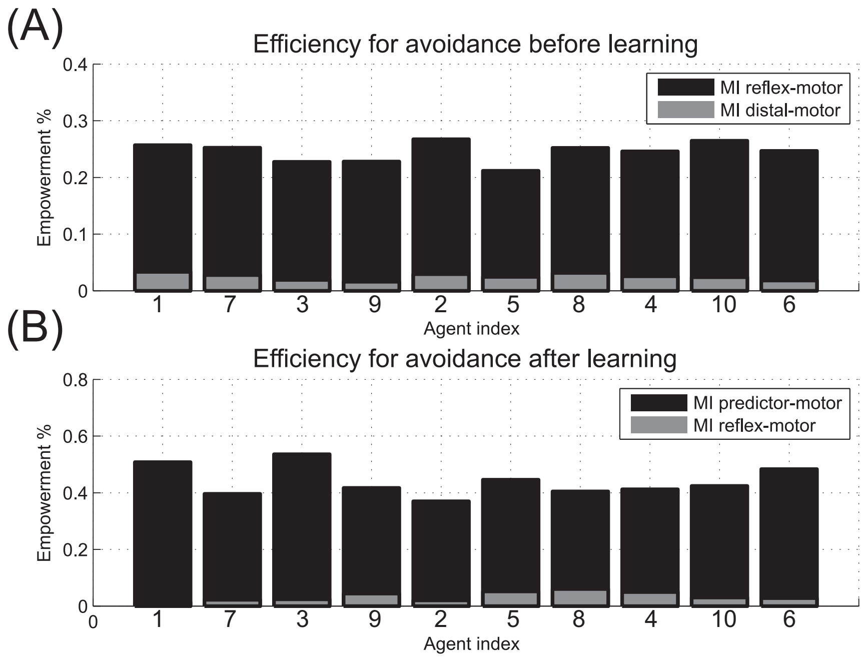


Figure 3.30: (A) Efficiency for every agent of the reflex-output and predictive-output loop in terms of capacity before learning (stage 1): $MI^4_{Av,U0}/\zeta^4_{all}\%$ (dark bars), $MI^4_{Av,U1}/\zeta^4_{all}\%$ (grey bars). (B) Efficiency after learning (stage 3).

3.3.7 Discussion

In summary, I have introduced an extension to Ashby's requisite variety theory called the law of adaptive requisite variety, computed the information flow to measure the learning performance for agents with competitive behaviours and found the relation between the efficiency of the information flow MI and the weight change of the adaptive controller $\Delta\omega_1$.

I also linked the information approach to the Luhmann theory that sub-systems are formed to reduce the perceived complexity of the environment. In my simulations, after the learning experience 5 agents have a dominant attraction behaviour for food disks (seekers) and 5 have a dominant attraction behaviour for others (parasites). The seekers mainly use the predictive information of the food disks while the parasites mainly use the predictive information of the others who posses food. Thus, my measure of information quantifies the information selection of the agents before and after learning which means I am able to discriminate which agent is a parasite or a seeker without looking at the value of the weights.

While Polani et al. 2004, 2005 and Pfeifer et al. 2008; Klyubin et al. 2008 used the empowerment measure as a general cost function to optimise the agent's behaviour or evolution, I use it as the upper bound of the MI to measure the efficiency of the sensory-motor loop use. Ay et al. (2008) use an adaptive controller which maximises the excess entropy (the mutual information between past and present) at the input side to achieve a working regime exploratory and sensitive to the environment. I can calculate the MI for this case by considering the reflex as the present input and the predictor as the past history. My approach is not restricted to MISO controllers. Kolodziejski and Kulvicius (2009) measures the temporal input development, the output and path entropy of the adaptive agents to study the optimality of the antenna ratio for an avoidance task, thus completing the tools required to evaluate a single task controller.

The following section 3.4 contains some experiment regarding the application of the mutual information to a Q-learning agent to verify that this approach is feasible also with reward based learning. After that, the section 3.5 introduces a mono-dimensional measure called Predictive Performance which summarises the learning performance with a singular scalar measure. This is necessary to avoid the multi dimensional analysis based on the mutual information of the predictor and reflex pathway: for every behaviour, like the food attraction, I have to look at both the values $MI^4_{Fo,U0}$,$MI^4_{Fo,U0}$. Things get more complicate when it is necessary to compare the performance of different agents, because then there is no normalisation basis for doing so. Another issue is the absolute performance in terms of regulation: how can I identify if one agent despite its efforts was able to keep its desired state. These questions will be answered in the section 3.5.

3.4 Introduction: information flow in Q-learning

This Section applies the information flow to an artificial agent which uses reinforcement learning for an obstacle avoidance task. The purpose of the experiment is to prove that the same principles introduced in the previous chapters apply to such a different on-line learning principle. I will introduce first reinforcement learning, then describe the robot-environment task with the learning controller and finally shows the result of the application of the information flow.

3.4.1 Methods: reinforcement learning

Reinforcement learning (Sutton and Barto, 1998) is characterised by a learning problem: an agent learns from its interactions with the environment to achieve a goal. Any method that is suited to solve this problem, is considered to be a reinforcement learning method. A reinforcement learning system is composed by:

- the agent: the learner and decision maker decides to make an action a_t
- the environment: what interacts with the agent. It is described by a state s_t and gives a reward r_{t+1} for each action.
- a policy π_t: a mapping from perceived states of the environment to actions to be taken when in those states
- a reward function: a mapping from perceived states (or state-action pairs) of the environment to a reward (a number). The reward defines what are the good and bad events for the agent.
- a value function V^{π}: a mapping from perceived states to a value which represents the expected total amount of reward that can be accumulated over the future.
- optionally a model of the environment

The agent and the environment interact in time steps $t = 0, 1, 2, 3, 4$ [6]. At each time step t, the agent produces a representation of the environment's state, $s_t \in S$, where S is the set of possible states and on that basis it takes an action $a_t \in A(s_t)$, where $A(s_t)$ is the set of available actions in the state s_t. One step later, the agent receives a numerical reward $r_{t+1} \in R$ and find itself in a new state s_{t+1}. At each time step the agent implements a mapping from states to probabilities of selecting each possible action. The probabilities are computed thanks to the agent's policy, where $\pi_t(s, a)$ is a mapping from each state s and action a to the probability of taking action $a_t = a$ when in state $s_t = s$. The vast majorities of reinforcement learning algorithms are based on estimating the value function that estimates how good it is for the agent to be in a given state. The positivity is defined in terms of future rewards. Thus, the value function $V^{\pi}(s)$, is the expected return when starting in s and following π thereafter:

$$V^{\pi}(s) = E(R_t | s_t = s) \tag{3.106}$$

$V^{\pi}(s)$ is the state-value function for policy π. There is also the complementary function Q^{π} called the action-value function for policy π: Q^{π} is the the expected return starting from s, taking the action a, and thereafter following policy π:

$$Q^{\pi} = E_{\pi}(R_t | s_t = s, a_t = a) \tag{3.107}$$

Reinforcement learning methods specify how the agent changes its policy as a result of its experience. This framework is quite flexible and can be applied to different scenarios: the states can be low-level sensations or they can be more abstracts like symbolic descriptions, the actions can be low level motor commands or high level decisions like mental choices. The

[6] for simplicity we assume a digital simulation but it can be extended to the continuous case see (Bertsekas and Tsitsiklis, 1996)

general rule to define the boundary between the agent and the environment is that anything that cannot be changed arbitrarily by the agent is considered to be the environment. The agent-environment boundary represents the limit of the agent's absolute control, not of its knowledge. Indeed the agent may know everything about its environment but the reward computation is out of the control of the agent because it cannot be influenced arbitrarily. The agent goal is to maximise the total amount of reward it receives in the long run, in the simplest case of an episodic task it is defined as:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T \tag{3.108}$$

where T is a final time step. An episodic task, like playing a chess game or solving a maze, is characterised by a terminal state, when the agent ends up in this state, the environment is reset to its starting state. If the goal requires a continuous-control, the final time T would be infinite and therefore we cannot maximise an infinite time series, therefore equation 3.108 needs to be modified as:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3.109}$$

where the parameter γ, $0 \leq \gamma \leq 1$ is called the discount rate. It determines the present value of future reward:

- if $\gamma = 0$, the agent maximises only immediate rewards and so it chooses a_t to maximise only r_{t+1}
- as γ approaches 1, the agent consider future rewards more important.

The value functions V^π and Q^π are estimated from experience. For example, if an agent follows the policy π and maintains an average, for each state encountered, of the actual returns that have followed that state, then the average will converge to the state's value, $V^\pi(s)$, as the number of times that state is encountered approaches infinity. If separate averages are kept for each action taken in that states, it will converge to the action values, $Q^\pi(s,a)$. The next section describes the connectionist Q-learning approach that will be used by the agent for an obstacle avoidance task.

3.4.2 Methods: Q-Learning algorithm

The Q-Learning algorithm suggested by Watkins in 1989 [1] is one of the most popular reinforcement learning algorithms. In Q-Learning the purpose of the agent is to find a control policy π which maximises the value function defined as:

$$V(s_t) \leftarrow E\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k} \tag{3.110}$$

$V(s_t)$ depends on the sequence of actions determined by the policy π. Q-Learning works on a Q-function which is computed from the value function in such a way:

$$Q(s_t, a_t) \leftarrow r_t + \gamma \cdot V(s_{t+1}) \tag{3.111}$$

where a_t is an action chosen at time t out of the set of possible actions A. Because the purpose of the system is to maximise the sum of total reward, $V(s_{t+1})$ is replaced by $max_{a \in A} Q(s_{t+1}, a)$ and thus the previous equation becomes:

$$Q(s_t, a_t) \leftarrow r_t + \gamma \cdot max_{a \in A} Q(s_{t+1}, a) \tag{3.112}$$

Q is a 2-D table where the rows contains actions and the columns contains states or vice-versa. When the state-action space is large (especially in continuous cases) more resources are required to store the table of evaluation. To solve those problems, the following approaches were introduced in the literature:

- discretisation of the Q table: Q table of large size is split into several Q tables of smaller size (Barto et al., 1990)
- Hamming distance approach
- CMAC method by Albus
- RBF similar to CMAC
- Neural Networks as suggested by Lin (Lin and Mitchell, 1992)

The following section describes the use of a a multilayer perceptron as a Q- learning table approximation. The joint use of MLP and the Q-learning algorithm is called connectionist Q-learning method.

3.4.3 Methods: Q-Learning connectionist

The tabular representation of the Q-function is replaced by a set of neural networks, each for every action. States are forwarded to the inputs of the neural network and outputs are the estimates of the Q-values. During each iteration of the learning algorithm, the current state of the system is forwarded to the inputs of each neural network, but the weights are only updated for the network whose action was selected. The weight correction error for the single step Q-learning is:

$$e_t = r_t + \gamma \cdot \max_{a \in A} Q(x_{t+1}, a_{t+1}) - Q(x_t, a_t) \tag{3.113}$$

The modified connectionist Q-Learning algorithm is summarised with the following pseudo code:

1. Set eligibility traces equal to zero, $e_0 = 0$
2. Initialize time at $t = 0$.
3. Select an action, a_t
4. If $t > 0$, update the weights
5. $w_t = w_{t-1} + \alpha \cdot ([r_{t-1} + \gamma \cdot \max_{a \in A} Q_t - Q_{t-1}] \cdot \nabla_w Q_{t-1} + [r_{t-1} + \gamma \cdot \max_{a \in A} Q_t - \max_{a \in A} Q_{t-1}] \cdot e_{t-1})$
6. $e_t = \nabla_w Q_t + \gamma \lambda e_{t-1}$
   Calculate the output gradient $\nabla_w Q_t$ only for the network whose action was chosen

7. Execute action a_t and receive reward r_t
8. If the absorbing state is reached, then stop; otherwise $t \leftarrow t + 1$ update time and go to Step 3.

3.4.4 Methods: the robot and the task

The robot is a Braitenberg vehicle as already described in Section 3.1.1 that can only execute 3 actions: move forward, turn left and right by a predefined angle. The robot is situated in a square arena where 20 obstacles are placed randomly for each session. The task of the robot is to minimise the number of collisions by learning appropriate motor responses from its sensory information. The sensory information is generated by an array of floor binary sensors numbered from 1 to 9 (see Figure 3.31) that detect the presence or absence of an obstacle on the world (binary input information). The robot receives a reward signal $r(t)$ at time t The geometric simulation parameters are described in the Appendix 5.2.9. The parameters for the MLP Q-learning algorithm are:

- the learning rate is $\alpha = 0.8$
- the forgetting rate for the eligibility traces is $\gamma = 0.8$
- the Q-learning factor $\lambda = 0.2$

The robot must receive a reward signal from the environment to be able to discriminate between good and bad actions. The reward structure was assigned in this way by the author:

- $r(t) = 0.1$ if at time t the robot has moved forward successfully
- $r(t) = -0.2$ if at time t the robot has collided with an obstacle
- $r(t) = 0$ if at time t the robot has collided with a wall of the world

For each simulation (see Figure 3.32) the robot goes through three different stages:

- in $T_0 = [0, 1 \cdot 10^6]$ the robot is purely reactive and does not learn by imposing $\lambda = 0.0$
- in $t = (1 \cdot 10^6, 2 \cdot 10^6]$ the robot is fully learning by imposing $\lambda = 0.8$
- in $T_\infty = (2 \cdot 10^6, 4 \cdot 10^6]$ the robot has learned from the previous section and now uses his weights w to avoid the obstacles.

3.4.5 Results: avoidance case

The information flow is calculated as in the previous Section 3.3, but in this case it is very easy because the output Z and the input S of the robot are already discrete. The input S is a binary word of 9 bits:

$$S = S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 \tag{3.114}$$

where each bit indicates if the sensor has touched the obstacle. For example the string $S =$ 100100000 indicates that the obstacle has collided with the input sensor in position 1 and 4 (see Figure 3.31). The output Z is encoded with a word of 2 bits:

$$Z = Z_1 Z_2 \tag{3.115}$$

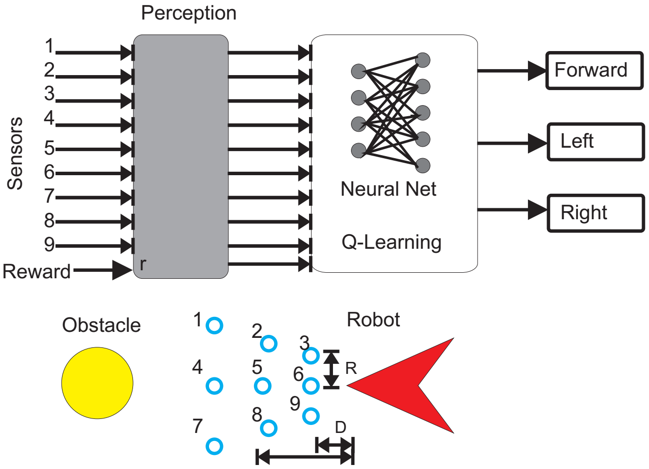


Figure 3.31: Simplified diagram of the q-learning robot. The yellow dots represent a binary input for the detection of the obstacle.

which encodes the 3 possible actions:

- move forward $Z_{fwd} = 00$
- turn left $Z_{left} = 01$
- turn right $Z_{right} = 10$

in this case 1 combination $Z_{null} = 00$ is not used because there are only 3 actions performed by the robot. To apply the information measure described in the Section 3.3, it is necessary to distinguish between the proximal or reflex signal X and the distal signal Y. The sensor inputs numbered 1,4,7 are far from the robot and thus can be regarded as distal inputs, whereas the inputs 3,6,9 are closer to the robot and thus can be regarded as proximal or reflex signals.

- $MI(X, Z)$ is the mutual information between the output Z and the reflex input X
- $MI(Y, Z)$ is the mutual information between the output Z and the distal input Y

X is thus encoded with a 3 bit word, Y is encoded as a 3 bit word and Z as described before as a 2 bit word. The information flow is computed from the previous quantities, by concatenating the output Z n times:

- $MI^n(X, Z)$ is the information flow of order n between the output Z and the reflex input X
- $MI^n(Y, Z)$ is the information flow of order n between the output Z and the distal input Y

I then calculated the reflex and distal information flow for the robot during the purely reflex phase and after learning. Table 3.7 shows the relevant values for a typical simulation run before and after learning:

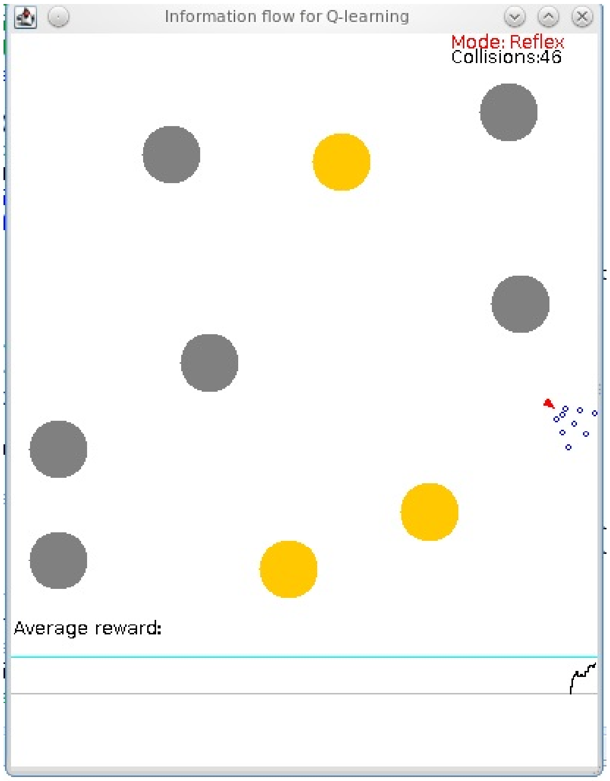


Figure 3.32: The obstacles are marked as yellow at the beginning. When an obstacle is touched it becomes orange to keep track of the collision history. The number of collisions and the average reward are shown in the simulation window. The red status label describes in what stage the robot is.

- because the agent has an instantaneous motor response, the order was computed for $n = 4, 5, 6$ actions
- R_0 indicates the average reward received by the robot before learning
- R_∞ indicates the average reward received by the robot after learning

Table 3.8 shows the same results for the same robot but with a 10% probability of choosing random actions during learning. This strategy increases the exploration probability and results in a better performance because $R_\infty = 0.0791 > 0.0688$ for the case without random selection. The clear result is that when the robot is learning is reducing the reflex information flow $MI(X, Z)$ and increasing the predictive information flow $MI(Y, Z)$, even though this behaviour was not designed but learned by the Q-learning approach. For example, if I consider the information flow of order 5 from Table 3.8, before learning the robot is using $MI^5(X, Z) = 1.0446$ bits in the reflex loop and only $MI^5(Y, Z) = 0.0376$ bits in the predictive loop but after learning, the robot is using less information from the reflex loop $MI^5(X, Z) = 0.0271$ and more information from the predictive loop $MI^5(Y, Z) = 1.8086$ bits. Same interpretation applies to the other cases described in the tables.

$\underbrace{Order}_{n}$	$\underbrace{MI(X,Z)}_{T_0}$	$\underbrace{MI(Y,Z)}_{T_0}$	R_0	$\underbrace{MI(X,Z)}_{T_\infty}$	$\underbrace{MI(Y,Z)}_{T_\infty}$	R_∞
4	0.0965	0.0864	0.0429	0.0102	0.3691	0.0688
5	1.0815	0.0864	0.0429	0.103	0.3747	0.0688
6	0.0965	0.0874	0.0429	0.0104	0.3824	0.0688

Table 3.7: Table containing the information flow when the robot is avoiding the obstacles without random selection of the actions.

$\underbrace{Order}_{n}$	$\underbrace{MI(X,Z)}_{T_0}$	$\underbrace{MI(Y,Z)}_{T_0}$	R_0	$\underbrace{MI(X,Z)}_{T_\infty}$	$\underbrace{MI(Y,Z)}_{T_\infty}$	R_∞
4	0.0521	0.0481	0.0473	0.0249	0.7861	0.0791
5	1.0446	0.0376	0.0473	0.0271	1.8086	0.0791
6	0.0523	0.0484	0.0473	0.0303	0.8240	0.0791

Table 3.8: Table containing the information flow when the robot is avoiding the obstacles with random selection of the actions at 10%.

The Tables 3.9 and 3.10 contain the statistical summary for 100 simulations computing the following values:

- average and standard deviation of the measure $MI(X, Z)_{T_0}/MI(X, Z)_{T_\infty}$
- average and standard deviation of the measure $MI(Y, Z)_{T_0}/MI(Y, Z)_{T_\infty}$
- average and standard deviation of the measure R_{T_0}/R_{T_∞}

The two cases with and withouth random selection of actions do not show a big difference in the reward, however in the case with random selection the mutual information flow is greater for the predictive pathway, suggesting that increasing exploration leads to a larger exploitation of the distal signal.

$\underbrace{Order}_{n}$	$\underbrace{MI(X,Z)}_{T_\infty/T_0}$	$\underbrace{MI(Y,Z)}_{T_\infty/T_0}$	R_∞/T_0
4	9.40 ±0.13	0.25 ±0.13	0.62 ±0.2
5	10.50 ±0.13	0.23 ±0.15	0.62 ±0.2
6	9.20 ±0.13	0.24 ±0.12	0.62 ±0.2

Table 3.9: Table containing the information flow for 100 simulations without random selection of the actions. Each value in the table contains the average ration and plus,minus the standard deviation of the ratio before and after learning.

$\underbrace{Order}_{n}$	$\underbrace{MI(X,Z)}_{T_\infty/T_0}$	$\underbrace{MI(Y,Z)}_{T_\infty/T_0}$	R_∞/T_0
4	2.12 ±0.13	0.66 ±0.12	0.61 ±0.2
5	38.54 ±0.14	0.58 ±0.15	0.61 ±0.2
6	1.78 ±0.13	0.62 ±0.11	0.61 ±0.2

Table 3.10: Table containing the information flow for 100 simulations with random selection of the actions. Each value in the table contains the average ration and plus,minus the standard deviation of the ratio before and after learning.

3.4.6 Discussion

The application of the information flow to this simple case of obstacle avoidance, gives an insight about the actual learning outcome of the robot. Before learning the robot uses a mixed combination of close and far sensors, whereas after learning the robot decreases the use of the close sensors for the benefit of the far ones. The initial use of the information flow is different for the ICO learning case which is programmed to use the reflex stimuli as a sort of wired behaviour. The Q-learning does not have any hard wired behaviour but only a random initialisation of the weights in the neural network and thus there is no preference between the stimuli. After learning the algorithm discovers that is not a good idea to react when the obstacle is too close even though the rewards are identical for a close or far collision. Although the computation of the information flow shows a clear development of behaviour from reactive to predictive, in ICO the reflex acts as the "reward" or the punishment channel in relation to Q-learning. Thus a more intuitive approach would have been to compute the two mutual informations $MI(R,Z)$ between output and reward as the reflex loop and $MI(Y,Z)$ between output and distal input. The main disadvantage of using this approach is that due to the sparsity of the reward signal it is necessary to use a more complex information model and was avoided to have a more behavioural based measure where we know how the agent is using its input differently. In summary by comparing the ICO approach with the Q-learning approach, there is similarity in terms of the strategy adopted by the robot in this particular scenario. It would be interesting to investigate if the similar property holds for different experimental setups where the task is not just attraction or avoidance. This investigation was not carried further in the Thesis as it was far away from the main topic but it is certainly something that can be investigated in the future.

3.5 The Predictive Performance measure

In the previous Sections, I have demonstrated that information flow measurements provide an index of how well each reflex and predictive pathway are used. In this Section, I will combine the information flow measurements with the reflex entropy to generate a single value which measures the learning performance. This scalar value is called Predictive Performance.

3.5.1 Introduction to closed loop measures

In my research study, I formulated a novel closed loop information measure - called predictive performance - which quantifies the learning performance of a line following robot. The robot is a classical Braitenberg vehicle (like the one described in Section 2.2.3) which has 2 retinal inputs functioning as far sensors and 2 small sensors acting as reflexes. The robot learns to follow tracks of different complexity by developing a retinal field using temporal sequence learning (ICO). I argue that measuring only the retinal weights (input) or the angular motion (output) provides a wrong estimate about the robot's adaptation to the track curvature. Thus an objective measure of the robot's performance -track deviation- is compared against the retinal field map and against the predictive performance measure. Simulations show that a robot with poor track performance has low retinal weights and a low predictive performance. Whereas a robot with a good track performance has high retinal weights and high predictive performance. However a robot with a poor track performance could have high retinal weights (I will explain why later), but the predictive performance will be low thus giving an objective measure of the performance. Therefore the predictive performance is a subjective measure of adaptation which reflects the objective performance of the robot. The measure can be extended to other types of adaptive predictive controllers.

Information measures are usually defined for input/output systems where they determine the quality of the transmission. Behaving agents, however, act as closed loop systems (see Fig.3.33) in which there is no clearly defined difference between input and output because the motor output influences the sensor input and so forth. What matters most for the organism is to compensate for disturbances P introduced by the environment into the perception action loop as in Fig.3.33(A). If there is no disturbance, the organism cannot differentiate between themselves and the environment. Consequently, the concept of information in these systems had to be revised (Porr and Wörgötter, 2005).

The new information measure called *Predictive Performance* is motivated by the theoretical foundation of the Radical Constructivism as described by Porr and Wörgötter (2005) stating that everything that every organism has a model of the environment described by his neural activity. In essence the controller acts as a reactive system before learning and as an open loop forward system after learning. In contrast to the previous work my new measure is independent of the learning rule and is not using its weights to compute the predictive performance. Instead I have employed a purely information theoretical approach.

I demonstrate my measure Predictive Performance in a simple robotics task where a robot has to learn to follow a line which is laid out with different curvatures so that different levels of difficulty can be evaluated. Learning drives the development of receptive fields in the robot similar to Kulvicius et al. (2007).

The Predictive Performance applied in this case, not only quantifies the relative performance of the robot for the 3 different tracks but also gives an index of the learning ability achieved

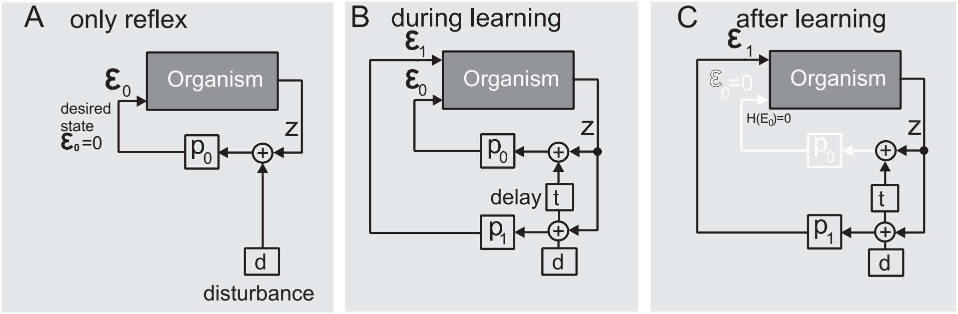


Figure 3.33: **A)** The organism is connected to the environment via the motor output Z and the reflex sensory input ϵ_0. The environment introduces a disturbance d via the transfer function p_0 which in turns change the reflex ϵ_0. The organism wants to keep the reflex to 0, so its desired state is ϵ_0.**B)** An organism can learn to keep its desired state by using a predictive input ϵ_1, providing that the disturbance d acts on the reflex ϵ_0 with a delay of t. **C)** After learning the organism should have reduced the reflex to 0 by using the predictive information ϵ_1.

by the robot for every single track. Numerical simulations show that the robot reduces the reflex information flow during learning and increases the retinal information flow if learning was effective. More specifically I show that an increase of Predictive Performance of a pixel in the receptive field is not equivalent to a high weight in the learning algorithm, which shows that one cannot rely on the open loop property to predict the performance of the agent but that instead it is necessary to use such a closed loop measure.

This rest of this section is divided as follows: setup of the robot, learning architecture, task and performance, symbols and convention used, application of the predictive requisite variety to a simple non learning robot, then to a full learning robot and finally the discussion.

3.5.2 Methods: experimental setup

The agent's task is to follow a black track of constant width but variable curvature in a 2 dimensional white arena. The agent is provided with a controller with a reflex steering behaviour which in most of cases (except of very shallow turns) will not be sufficient to steer the curve. As a consequence the robot looses the track. The learning goal is thus to learn predictive and smoother steering reactions in order to stay on the track and to avoid the initial reflex.

The reflex is generated by using two pixels close to the bottom of the robot's visual field are are called x_0^L and x_0^R which generate a difference signal ϵ_0 which is used as the reflex signal for steering and also drives learning of the receptive fields.

The agent uses 2 predictive receptive fields $x_{1,i,j}^L$ and $x_{1,i,j}^R$ for the left and right eye respectively. The receptive fields have a size of $N_{rf} \times N_{rf}\ px$ (Fig. 3.34 A) where each pixel within the receptive field represents an individual input $x_{1,i,j}$. The left and right pixel intensities are

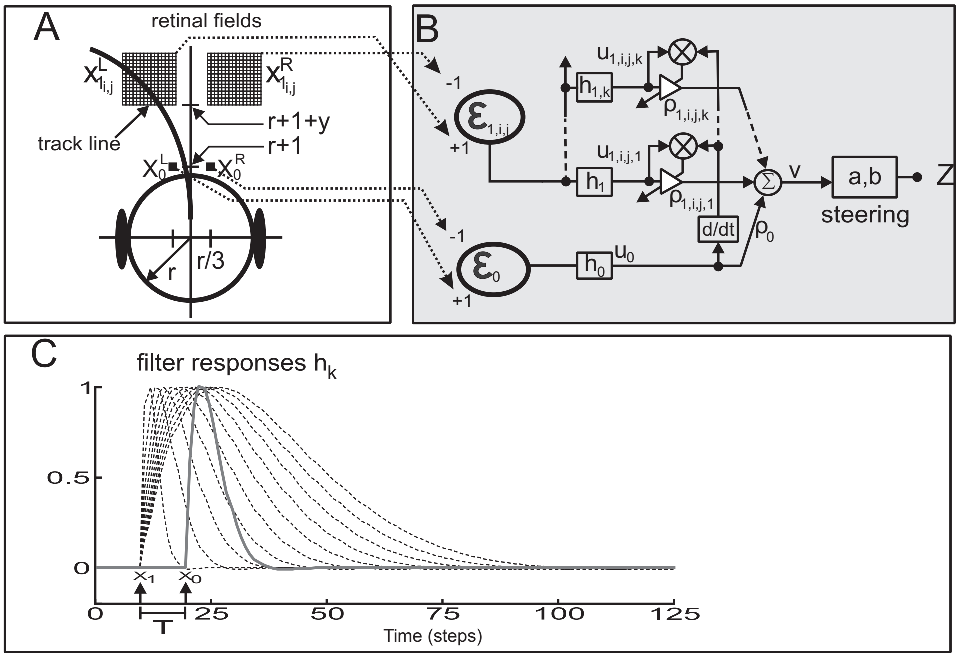


Figure 3.34: Physical and neuronal setup of the receptive field (RF) development using the simple learning architecture. **A)** Left and right retinal fields. The receptive filed positions are denoted by $x^{L,R}_{1_{i,j}}$, where $i = 1 \dots N_{rf}, j = 1 \dots N_{rf}$ are the indices of the RF pixels, and sensor field positions $x^{L,R}_0$. **B)** The simple neuronal setup of the robot. Symbols u denote filtered input signals x, ρ connection weights and v the output of the neuron used for steering. v is calculated by the method shown in C and its corresponding Eq. 3.121 given in Eq. 3.123 and transforms v to the motor output. a is the acceleration gain and b is the braking gain. Schematic diagram of the learning system. Inputs x, resonator filters h, connection weights ρ, output v. The symbol $\otimes$ denotes a multiplication, d/dt a temporal derivative. The amplifier symbol stands for a variable connection weight. Dashed lines indicate that input x_1 is fed into a filter-bank. **C)** Resonator filters h_0 (solid line) for the input signal x_0 and $h_{1,k}$ (dashed lines) for the x_1 given by parameters $f_{1,k} = 2.5/k\ Hz$, $k = 1, \dots, 10$ for the filter-bank in the x_1 pathway. Frequency of the x_0 pathway was $f_0 = 1.25\ Hz$. Damping parameter of all filters was $Q = 0.6$.

then combined into left and right difference signals pixel by pixel:

$$\epsilon_0 = x^L_0(t) - x^R_1(t) \quad (3.116)$$

$$\epsilon_{1,i,j}(t) = x^L_{1,i,j}(t) - x^R_{1,i,j}(t) \quad (3.117)$$

All difference signals are then filtered by low pass filters

$$\begin{aligned} u_0(t) &= h_0(t) * \epsilon_0(t) && (3.118) \\ u_{1,i,j,k}(t) &= h_{1,k}(t) * \epsilon_{1,i,j}(t) && (3.119) \end{aligned}$$

where $h_0(t)$ and $h_{1,k}(t)$ are low-pass filters which I define by its impulse response:

$$h(t) = \frac{1}{\beta} e^{\gamma t} \sin(\beta t), \tag{3.120}$$

where, $\gamma = -\pi f/Q$ and $\beta = \sqrt{(2\pi f)^2 - \gamma^2}$, with f the frequency and $Q > 0.5$ the damping. The index k in Eq. 3.119 denotes a filter bank for the predictive inputs so that every pixel of the receptive field is fed into $1 \ldots K$ filters. This filter bank will be used for learning which is described in the next section.

The filtered signals Eq. 3.118 and Eq. 3.119 are then fed into a summation unit where every signal have a weight associated to it:

$$v = \rho_0 u_0 + \sum_{i,j,k} \rho_{1,i,j,k} u_{1,i,j,k} \tag{3.121}$$

where v is is the steering angle which is calculated as the robot's position in the cartesian bi-dimensional space $(S_x(t), S_y(t))$.

The robot has a constant speed of 1 Unit/s

$$\begin{aligned} s(t) &= \Theta\left[speed - |v| \cdot b\right] && (3.122) \\ z(t) &= z(t-1) - v(t) \cdot a && (3.123) \end{aligned}$$

where a is the acceleration gain $a = 0.02$, b is the breaking factor $b = 0.005$ and Θ is the heaviside function. a controls the angular speed of the robot whereas b simulates breaking during turning.

$$\begin{aligned} S_x(t) &= S_x(t-1) + s(t) \cdot \cos(z(t)) && (3.124) \\ S_y(t) &= S_y(t-1) + s(t) \cdot \sin(z(t)) && (3.125) \end{aligned}$$

where $z(t)$ and $s(t)$ are computed in the previous equation Eq.3.123.

3.5.3 Methods: learning algorithm

The temporal sequence learning rule was used again for learning (Porr and Wörgötter, 2006). The general scheme of such learning algorithm is presented in Fig. 3.34 C.

Weights change according to an input-input correlation (ICO) rule :

$$\dot{\rho}_{i,j,k} = \mu u_{i,j,k} \dot{u}_0, \quad j > 0, \tag{3.126}$$

which is a modification of the isotropic sequence order (ISO) learning rule (Porr and Wörgötter, 2003). The behaviour of this rule and its convergence properties are discussed in (Porr and Wörgötter, 2006). The indices i and j denote the different pixels and the index k is the filter number of the filter bank.

The filter bank is needed to establish a temporal overlap of the signals from the receptive field and the reflex as demonstrated in Fig. 3.34C. Remember that the sensor fields $x_0^{L,R}$ are located at the bottom whereas sensor fields $x_{1,i,j}^{L,R}$ are placed higher up from the reflex.

The time delay T between the predictive receptive field $x_{1,i,j}$ and the reflex $x_0^{L,R}$ depends on the speed of the robot and direction angle with respect to the curvature. As shown in older studies of Porr and Wörgötter (2003); Porr and Wörgötter (2006), the number of filters K is not critical and here $K = 10$ was used. The simulated robot has a speed of 1 $Unit/s$ with filter coefficients $f_{k=1...K} = 0.1k$, for the filter-bank in the $x_{1,i,j,k}$ pathway. The frequency of the $u_0 = h_0 * x_0$ pathway was $f_0 = 1.25\ Hz$. Damping parameter of all filters was $Q = 0.6$.

3.5.4 Methods: symbols and conventions

Before introducing the new measure Predictive Performance, it is necessary to define the notation, symbols and information measures on which it is built. I am first defining the used symbols, then introduce the necessary information measures and finally, I will define my predictive performance.

Symbols and conventions

The symbols used in this section follows the convention:

- capital letters such as X indicate a random discrete variable.
- the symbols of the random variable are indicated by the set $X = \{x_1, x_2, ..., x_S\}$ where S is the number of symbols in this set.
- non capital letters such as x indicate the corresponding discrete time series $x(k)$ from which I estimate the density $P(X)$ of the corresponding random variable X.
- a time series is an ordered sequence of symbols $x(t) = \{x(0), x(1), ..., x(t)\}$ for $t \geq 0$
- the estimated entropy of a random discrete variable X is identified by $H(X)$ and is measured in bits.

I then identify my controller with the following variables and measures:

- E_0: random variable for reflex input
- $E_{1,i,j}$: random variable for predictive input located at the retinal coordinate i, j
- Z: random variable for motor output
- $H(E_0), H(E_1), H(Z)$: entropy of the random variables E_0, E_1, Z
- $I(X, Y)$= mutual information between variable X and variable Y
- $MI^n(X, Y, \tau)$= mutual information of order n between X and the delayed version of Y by factor τ (see Appendix 3.5.4 for more details)
- $MI(X, Y) = MI(X, Y, \tau = 1)$

In the following subsections I am going to describe the different information measures which can be applied to the robot. I show that these measures by themselves are not a good measure for the performance of the agent but combined will provide a measure which reflects the performance of the agent in a normalised way. Initially I introduce the information measures, then I apply it in the closed loop and finally I combine to form the Predictive Performance.

Input reflex entropy

The entropy $H(E_0)$ is the uncertainty of the reflex input or the average description necessary to encode the reflex input D_0. Before I am going to look at the actual values I need to recall that the reflex input is part of the reflex loop (see Fig. 3.33) via the differences of $x_0^L - x_0^R$, v_0, z, the environment p and back to ϵ_0. This loop is disturbed by the perturbation d which is then eliminated by the loop. Remember that ϵ_0 is essentially an error signal which has to be kept close to zero which is only the case for Eq. 3.128. However, because it is a reflex loop the feedback loop always reacts too late so that the input ϵ_0 can never reach a constant value. Thus, the entropy at ϵ_0 reflects the entropy originating from d. In my setup the reflex has an alphabet of 3 symbols $E_0 = \{-1, 0, 1\}$ to encode the line position on the left, right or center. The condition $H(E_0) = 0$ of null entropy, indicates that the closed loop controller has achieved perfect regulation, however there are 3 possible conditions with equal null input entropy:

$$\epsilon_0(t) = \{1, 1, 1, 1\} \rightarrow H(E_0) = 0 \tag{3.127}$$

$$\epsilon_0(t) = \{0, 0, 0, 0\} \rightarrow H(E_0) = 0 \tag{3.128}$$

$$\epsilon_0(t) = \{-1, -1, -1, -1\} \rightarrow H(E_0) = 0 \tag{3.129}$$

I can exclude condition Eq. 3.127 and Eq. 3.129 because they only arise when the feedback loop has failed totally. In order to have a successful learning, it is required at least a working feedback loop (Porr and Wörgötter, 2005) as a starting point. A perfect organism would have an input sequence identical to Eq. 3.128, however due to the causal nature of the sensory-motor loop there will be always a delay between the reaction and the disturbance as in Eq. 3.130 and Eq. 3.131. Thus there is never zero entropy at ϵ_0 as long as the reflex keeps the robot on track:

$$\epsilon_0(t) = \{-1, 0, 1, 0, -1, 1\} \rightarrow H(E_0) = 1.6 \tag{3.130}$$

$$\epsilon_0(t) = \{-1, -1, 0, 0, 1, 1\} \rightarrow H(E_0) = 1.6 \tag{3.131}$$

the entropy reaches a maximum of 1.6 bits which means that the input has a uniform distribution and thus is not constant in time. The input entropy does not tell us anything about the effort that the controller is doing to keep its desired goal, because it might be that the robot is not moving at all or that the environment is very simple (e.g. straight line). For that reason I need to define now truly closed loop measures.

Retinal predictive entropy

Remember that the goal of the learning algorithm is to make the reflex pathway redundant. In order to achieve this it aims to predict the trigger of the reflex via the reflex input (see Eq. 3.116) with the predictive inputs of originating from the difference of the receptive fields (see Eq. 3.117).

I define $H(E_{1,i,j})$ as the uncertainty of the predictor inputs at the retinal position (i,j), whereas

$$H(E_1) = \frac{1}{N_{rf}^2} \cdot \sum_{i,j=1}^{N_{rf}} H(E_{1,i,j}) \tag{3.132}$$

is the average entropy of the retinal differential input.

Mutual information

The previous sections described only input measures but I also need additional measures calculated between the output and the input of the agent to measure the information flow in the closed loop. The mutual information takes into account the performance of the controller by measuring how the controller reacts to the error signals at the reflex and predictive inputs:

1. $MI(Z, E_0, \tau, n)$: mutual information of the reflex loop

2. $MI(Z, E_{1,i,j}, \tau, n)$: mutual information of the predictive loop for the single input $E_{1,i,j}$

3. $MI(Z, E_1, \tau, n)$: average mutual information of the predictive loop which is computed as:

$$MI(Z, E_1) = \frac{1}{N_{rf}^2} \cdot \sum_{i,j=1}^{i,j=N_{rf}} I(Z, E_{1,i,j}, \tau, n) \tag{3.133}$$

where the parameter τ is the temporal difference between the motor output z and the sensory input $\epsilon_0, d_{1,i,j}$ and n is the sensory integration window. For instance when computing $MI(Z, E_0, \tau, n)$, I am considering as the random variable Z the motor output $z(k)$ and as random variable E_0 the sensory input sequence $d_0(k+\tau), d_0(k+\tau+1), ..., d_0(k+\tau+n-1)$. When τ and n are omitted that indicates that the mutual information has been maximised over the 2 parameters:

$$MI(Z, E_{0/1,i,j}) = \max_{\tau,n} MI(Z, E_{0/1,i,j}, \tau, n) \tag{3.134}$$

The mutual information $MI(Z, E_0), MI(Z, E_{1,i,j})$ is a measure of the *controllability* of the agent. To demonstrate this I show the two extreme cases:

- if $MI(Z, E_0) = 0$ then Z and E_0 are independent. It means that there is no correlation between the actions and the inputs of the robot. Imposing a motor value does not give a desired input. For example the series:

$$\begin{aligned} z(t) &= \{1, 2, 3, 4, 5, 6\} && (3.135) \\ \epsilon_0(t) &= \{-1, 0, -1, 1, -1, 0, 1\} && (3.136) \end{aligned}$$

  have null mutual information $MI(Z, E_0, \tau = 1) = 0$ bits.

- if $MI(Z, E_0) = max(H(Z), H(E_0))$ then Z and E_0 are perfectly dependent. It means that this time when the robot imposes a motor action it will have a better chance of reading a desired input. For example if $MI(Z, E_0) = 1.6$ bits, then the robot can choose a motor action and read a desired input in an average run. But if the robot looses 0.6 bit and goes to $MI(Z, E_0) = 1.0$ then in the average run 1 particular motor action will yield 2 equiprobable inputs at E_0 and thus the robot has less control over its environment.

An equivalent description of the mutual information can be done with the conditioned entropy:

$$\begin{aligned} MI(Z, E_0) &= H(E_0) - H(E_0|Z) && (3.137) \\ MI(Z, E_{1,i,j}) &= H(E_{1,i,j}) - H(E_{1,i,j}|Z) && (3.138) \end{aligned}$$

Here $H(E_0|Z)$ and $H(E_{1,i,j}|Z)$ are the conditional entropies. Since $H(E_0) \geq H(E_0|Z)$, this characterization is consistent with the non negativity property of entropy. If entropy $H(E_0)$ is regarded as a measure of uncertainty about a random variable, then $H(E_0|Z)$ is a measure of how much the motor output Z has no influence over the input E_0. This is the amount of uncertainty remaining about E_0 after Z is chosen, and thus the right side of the first of these equalities can be read as the amount of uncertainty in E_0, minus the amount of uncertainty in E_0 which remains after Z is chosen, which is equivalent to the amount of uncertainty in E_0 which is removed by imposing Z. In a broader sense the mutual information $MI(Z, E_0)$ measures the quantity of information that the robot is able to recover from its inputs given its outputs. In control theory I can think about the $MI(Z, E_0)$ as a measure of the controllability: if it is maximum it means the robot can reach any desired input by a determined action, if not the robot cannot reach certain desired states with absolute certainty.

However, the mutual information via the reflex or the predictive pathway itself is not sufficient as a performance measure. Remember that I would like to measure the success of learning, especially if it is using the predictive pathway via $\epsilon_{1,i,j}$ to eliminate the pathway via ϵ_0. In other words I need to verify if the mutual information is transferred from the reflex to the predictive pathway *and* that the error of the reflex has been reduced to $\epsilon_0 = 0$ after learning. I combine these requirements in one measure called "predictive performance" which I am describing in the next section.

3.5.5 Methods: Predictive Performance measure

The predictive performance measure is computed by considering the information measures introduced before. The subscript $t = 0$ and $t = \infty$ indicate respectively the measure computed before learning and after learning after the weights have been stabilised. Table 3.11 contains the four values that are relevant to the agent's performance:

Table 3.11: Information measures used for the computation of Predictive Performance.

	$H(E_0)$	$MI(Z, E_{1,i,j})$	$MI(Z, E_0)$
before learning	$H(E_0)_{t=0}$	$MI(Z, E_{1,i,j})_{t=0}$	$MI(Z, E_0)_{t=0}$
after learning	$H(E_0)_{t=\infty}$	$MI(Z, E_{1,i,j})_{t=\infty}$	$MI(Z, E_0)_{t=\infty}$

The predictive performance is then computed as:

$$PP_{i,j} = \frac{H(E_0)_{t=0} - H(E_0)_{t=\infty}}{H(E_0)_{t=0}} \cdot \frac{MI(Z, E_{1,i,j})_{t=\infty}}{MI(Z, E_0)_{t=0}} \qquad (3.139)$$

The two factors of this equation need to be discussed now:

- The first factor of Eq. 3.139 provides a measure reflecting the reduction of entropy of the error signal E_0 which drives the reflex. Remember that the goal of learning is to avoid the reflex which in an ideal case will lead to no trigger of the reflex or $\epsilon_0 = 0$. In a realistic scenario the reflex entropy will decrease but the never reach zero because the agent might do mistakes from time to time. In general the entropy should be lower after learning than before: $H(E_0)_{t=0} \geq H(E_0)_{t=\infty}$. Thus, the first factor in Eq. 3.139 will be one for perfect avoidance of the reflex and zero for no change in the reflex entropy.

- The second factor makes sure that the agent controls its own actions before $(MI(Z, E_0)_{t=0})$ and after $(MI(Z, E_{1,i,j})_{t=\infty})$ learning. This factor makes sure that before and after learning the agent is able to generate actions which control its own inputs. Remember that before learning this is done via the reflex input and that this is my starting point. After learning the agent should be in control of its own actions via the predictive inputs. Ideally, these two mutual information values should be similar which means that control before and after is guaranteed. Or in other words, the controllability should be transferred from the reflex $(MI(Z, E_0)_{t=0})$ to the predictor $(MI(Z, E_{1,i,j})_{t=\infty})$.

An example of a perfect learner can be an agent with the following values:

	$MI(Z, E_0)$	$MI(Z, E_{1,i,j})$	$H(E_0)$
before learning	**2.5 bits**	1.5 bits	**3 bits**
after learning	0	**2.5 bits**	**0 bit**

which results in a Predictive Performance of $PP = 1$ (see Eq. 3.139). The important values here are in bold. The mutual information is completely transferred from the reflex pathway to the predictive pathway and the error in ϵ_0 is reduced to zero bits. In fact a necessary but not sufficient condition for learning is that $MI(Z, E_0)_{t=0} \cong MI(Z, E_{1,i,j})_{t=\infty}$ because the predictor should be able to provide information that has to be learned or exploited by the controller. The following section describes behavioural experiments conducted to demonstrate the predictive performance.

3.5.6 Results: behavioural experiments

Task

The task of the robot is to follow a track. I designed tracks of increasing difficulty. There are three simple tracks (see Fig.3.35(A)) with an increasing curvature ratio and a complicate track (see Fig.3.35(B)) with left and right turns of different curvatures. The retinal field that will be learned will have different structure for every track and the robot will show a different level of performance.

Behavioural performance

In order to assess the Predictive Performance, it is necessary to introduce an objective or behavioural performance. Learning is successful when the robot is not triggering the reflex any more, thus minimising its distance from the track. Thus the deviation from the track is simply defined as the average deviation of the robot's position (defined by the mass centre of the robot) from the track. It is obtained from the robot's driving trajectory and is calculated

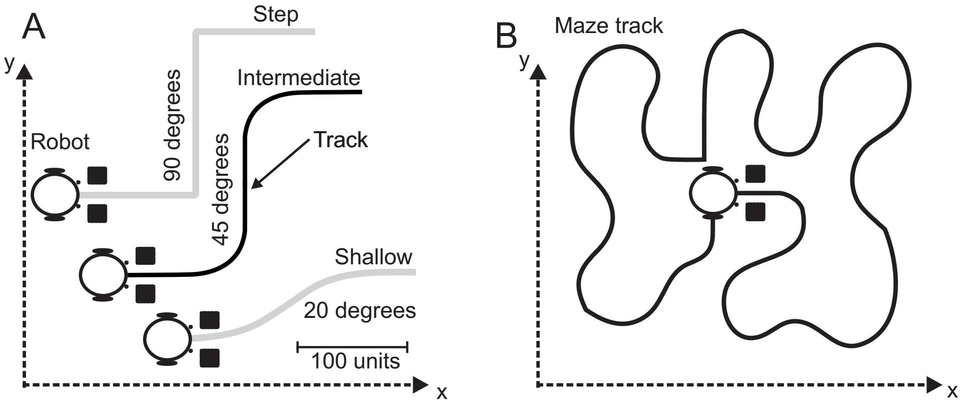


Figure 3.35: **A)** Three tracks of increasing difficulty: shallow, intermediate and step. The Cartesian coordinate system has origin in the bottom left corner. **B)** A maze track with different curvatures, the robot starts in the middle.

by the Euclidean distance:

$$\psi = \frac{1}{N_{sim}} \sum_{t=0}^{N_{sim}-1} \sqrt{(S_x(t) - x_t(t))^2 + (S_y(t) - y_t(t))^2} \; units, \qquad (3.140)$$

where $S_x(t)$ and $S_y(t)$ are the coordinates of the robot's position at time moment t, $x_t(t)$ and $y_t(t)$ are the coordinates of the track point from which the distance to the robot's position is minimal, and $t = 0 \ldots N_{sim} - 1$ denote the driving duration and is measured in simulation steps. This measure ψ will be used as an objective measure of the robot's performance for each track type.

A simple non-learning robot

Before applying the Predictive Performance to a complex retinal based robot, it is better to measure it on a simplified robot with few parameters and an intuitive behaviour. The simplified robot is described in Fig.3.36 and to its parameters k_1, k_2, it is possible to show how the predictive performance works when the robot switches between the reflex pathway ϵ_0 to the predictive pathway $\epsilon_{1,i,j}$ by manually setting the gain of the reflex and predictor. The reflex x_0 here is a digital b/w line sensor and the predictor $x_{1,i,j}$ is a b/w retina of 4x4 pixels. The difference between the left and right reflex ϵ_0 is fed into a band pass filter to produce u_0. The difference between the left and right retina $x_{1,i,j}$ is fed into 16 band pass filters to produce $u_{1,i,j}$. The distance between the reflex and the predictor is y and will assume the following values ($y = \{6, 12, 24\}$ units) to do a comparative analysis of the performance.

The neuronal output is computed as the synaptic input summation of the reflex and pre-

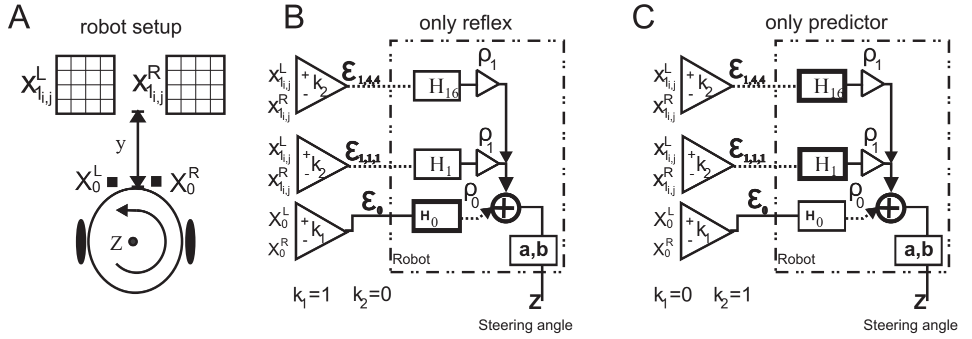


Figure 3.36: **A)** A simple non learning robot with a 4x4 retinal field. **B)** In this configuration the robot uses only its reflex ϵ_0 and ignores -while still experiencing- the predictor $\epsilon_{1,i,j}$. **C)** In this configuration the robot uses only its predictor $\epsilon_{1,i,j}$ and ignores -while still experiencing- the reflex ϵ_0.

dictor:

$$\begin{aligned} v(t) &= k_1 \cdot u_0 + k_2 \sum_{i,j}^{i,j=4} u_1 \\ \mu &= 0 \end{aligned} \tag{3.141}$$

Where the parameters k_1, k_2 can be set to only test the reflex pathway when $k_2 = 0$ and test the predictor pathway when $k_1 = 0$. The robot is not learning anything because the learning speed is set to zero -$\mu = 0$- thus the weights are steady. The parameters $k_1, k_2, a, b, \rho_{1,i,j}$ are tuned manually so that the robot is able to complete the track. The predictive information flow here is a simple average over the individual retina channels because these channels have no specific meaning as they have been chosen manually. The predictive information flow becomes:

$$MI(Z, E_1) = \frac{1}{16} \sum_{i,j=1}^{i,j=4} MI(Z, E_{1,i,j}) \tag{3.142}$$

indicating that Eq.3.139 for the PP is essentially reduced to a single pixel for the reflex and a single pixel for the predictor.

Table 3.12 contains the computed parameters for the PP measure in the maze track scenario. The first column shows the different distances y between the predictor and the reflexes in pixels which is used here to vary the level of difficulty for learning. I am going to argue that both fractions in Eq.3.139 are needed to combine the measures which include mutual information with the measure of the reflex entropy.

First of all the reflex input entropy $H(E_0)$ is the standard measure to evaluate the performance of a closed loop controller: the entropy is zero when the desired state is achieved ideally after learning. The sixth and seventh column of Table 3.12 contain the input entropies of the reflex before $t = 0$ and after $t = \infty$ learning. When the distance is $y = 6, 10$ the reflex pathway is totally removed $H(E_0)_{t=\infty} = 0$, whereas for the distance $y = 12$ there is a remaining error of $H(E_0)_{t=\infty} = 0.00536$, indicating that in very few occasions the robot sill

Table 3.12: Table measuring the predictive performance for different distances y of the predictor as seen in Fig.3.36(C).

y	$\underbrace{MI(Z,E_0)}_{t=0}$	$\underbrace{MI(Z,E_1)}_{t=0}$	$\underbrace{MI(Z,E_0)}_{t=\infty}$	$\underbrace{MI(Z,E_1)}_{t=\infty}$	$\underbrace{H(E_0)}_{t=0}$	$\underbrace{H(E_0)}_{t=\infty}$	PP
6	4.16 bits	3.99 bits	0	3.7 bits	0.137551	0	0.89
10	4.16 bits	4.12 bits	0	3.91 bits	0.137551	0	0.939
12	4.16 bits	4.17 bits	0.0453	3.91 bits	0.137551	0.00536	0.902

needs to use its reflex even when the predictive pathway is performing most of the steering reaction. This obviously contributes to a lower Predictive Performance in the last row of the table. Nevertheless, a zero input entropy at ϵ_0 does not mean that learning has been successful. For example, the robot could have just driven into an easier section of a track like a straight line where the absence of steering does not trigger any reflex. This issue can now be solved by considering the mutual information between output and inputs (see columns 2-5 in Table 3.12). The first column shows the mutual information of the reflex pathway $MI(Z,E_0)_{t=0}$ before learning which is identical for all cases. The next column shows the mutual information of the predictive pathway $MI(Z,E_1)_{t=0}$ before learning that provides whether or not learning is possible. A non-zero value indicates that the predictor is highly correlated with the output z and therefore learning is possible. The next two columns contains the mutual information for the reflex and predictive pathway after learning: a robot with an uncorrelated behaviour will generate very low values for mutual informations. An agent which has learned to use the predictive pathway will generate instead high values for the mutual informations.

It is impossible to use only the mutual information of the reflex loop after learning to determinate the performance because, for example when the distance is $y = 6, 10$ $MI(Z,E_0)_{t=\infty} = 0$. This condition would indicate a total loss of control in the robot but the interpretation is ambiguous because the reflex entropy $H(E_0) \simeq 0$ is almost null because ϵ_0 is converging to zero. This basically means that the reflex pathway after learning $MI(Z,E_0)_{t=0} \simeq 0$ becomes unused.

However there is an important property that has to be observed: when the agent transfer its control from the reflex pathway to the predictive pathway the mutual information has to be transferred as well from the reflex pathway $MI(Z,E_0)_{t=0}$ to the predictive pathway $MI(Z,E_0)_{t=\infty}$. Table 3.12 clearly show how the initial information flow of the reflex pathway which is about 4.16 bits is transferred to the information flow of the predictor pathway almost intact (just a little less than 4 bits) for the cases $y = 10, 12$. For the short configuration $y = 6$ the reduction was from 4.16 to 3.7 bits but this is due to the manual setup of the weights.

This is the reason why the predictive performance uses both the input entropy $H(E_0)$ and the mutual information $MI(Z,E_0)_{t=0}$ before learning and after learning $MI(Z,E_0)_{t=\infty}$.

The last column of Table 3.12 contains the predictive performance computed for each parameter setting of the y distance. The maximum value is achieved when $y = 10$ because there is a good information transfer from the reflex to the predictor pathway $4.16 \rightarrow 3.91$ bits and a null input entropy after learning $H(E_1) = 0$. For $y = 12$ the performance is lower because the input entropy after learning $H(E_1) = 0.00536$ the input entropy is bigger then 0. The worse performance is for $y = 6$ because of the lower information transfer $4.16 \rightarrow 3.7$ but with null input entropy after learning $H(E_1) = 0$. This shows that minimizing the reflex entropy it not sufficient to generate a controllable robot.

The best performance $PP = 0.939$ is achieved when the predictor-reflex distance is at $y = 10$, but how does it relates to the real objective performance of the robot? Table 3.13 provides the answer by computing the track deviation when the robot is following the track. For instance when the PP is maximum for $y = 10$ the track deviation $\Psi \simeq 4.038$ is minimum, whereas for the worse performance when $y = 6$ the PP is minimum and the track deviation is maximum $\Psi \simeq 4.95$. This relationship between PP and Ψ indicates that the predictive performance is well correlated with the objective track performance of the robot. The first row in Table 3.13 also measures the track deviation when the robot is only using the reflex $\Psi \simeq 1.93$ and shows that after the robot switch to the predictive pathway the track deviation is bigger rather then smaller as one would expect for a learning robot. Essentially a purely reflex based robot has the best performance $\Psi \simeq 1.93$ compared to the only predictor based case. This is because I have chosen the gain of the predictor field to an arbitrary gain by "hand" that produces over steerings even though the robot is always on track. Because the robot's retinal weights are manually set, it was not possible to achieve a better performance but for an autonomous learning robots learning must provide a better track deviation.

Table 3.13: Table measuring the track deviation Ψ for the simple robot as seen in Fig.3.36(B),(C) on the maze track.

Mode	Ψ
only reflex	1.936951
only predictor at $y = 6$	4.959343
only predictor at $y = 10$	4.038652
only predictor at $y = 12$	4.119643

The adaptive receptive field solves exactly this problem: setting the gain of every pixel in the retinal field to minimize the reflex error. In the following sub sections I am going to show what happens in the learning case.

A learning robot

In this section the predictive performance for the learning robot described in Figure 3.34 is computed. The robot has a retinal field of 15x15 pixels (see section 3.5.2), where each pixel has a weight and a set of filter banks so that the retina can develop a receptive field. The PP values are computed for each pixel and then compared to the synaptic weights of the receptive fields. This section is divided in two sub cases:

- the predictive performance is computed as in the simplified case before and after learning for the three standard tracks
- the predictive performance is computed during learning during the maze track

Predictive Performance after learning The robot completes each track NT_{before} times before learning (only reflex behaviour $\mu = 0$) and NT_{after} times until weights $\rho_{1,i,j,k}$ are stabilised ($\mu > 0$). The values NT_{before}, NT_{after} are different for each track because generally speaking a more complex trajectory will required a longer stabilisation period. The following list contains the observed values for each track:

1. for the shallow track $NT_{before} = 22$,$NT_{after} = 10$.
2. for the middle track $NT_{before} = 22$,$NT_{after} = 9$.
3. for the step track $NT_{before} = 22$,$NT_{after} = 17$.
4. for the maze track $NT_{before} = 1$,$NT_{after} = 1$.

The learning speed is set to $\mu = 0.5 \cdot 10^{-8}$, there are $K = 10$ bank filters for each pixel input and the distance between the retinal field and the reflex sensor is set to $y = 6$. There is no need to freeze the ICO weights because learning quickly stabilises during the experiment. Simulations are run with the learning robot and the following conditions:

1. for the shallow track in Figure 3.37.
2. for the middle track in Figure 3.38.
3. for the step track in Figure 3.39.
4. for the maze track in Figures 3.40,3.41.

Figures 3.37,3.38,3.39 contain 4 panels:

A The retinal field contains the learned weights $\rho_{1_{i,j}}$ averaged over the $n = 10$ filters for each retinal pixel within the coordinate $i = 1, ..., N_{rf}$,$j = 1, ..., N_{rf}$

B The retinal flow $MI(Z, E_{1,i,j})$ computed when the robot is only using the reflex E_0 for every pixel (i, j)

C The trajectories produced by the robot during learning. The small inset shows the average weight development of the retinal field $TOTAL_{RF} = \sum_{i,j,k} \rho_{1_{i,j,k}}$ over time.

D The predictive performance $PP_{i,j}$ is computed after the weights have been stabilised. The values of the reflex input entropy used to compute the $PP_{i,j}$ for each track are summarised in Table 3.14.

It is time to compare the weight development and the Predictive Performance for each track from the shallow to the step one. The main message is that for the easier track the weights and the Predictive Performance are nearly identical whereas for the most difficult track there is a substantial difference because high weights does not mean better performance.

$H(E_0)_{t=t_L} = 0$ means that the reflex is successfully eliminated in every case once learning is stable. This indicates that apparently the robot performs well in each track type, but there is a substantial difference in the retinal flow and hence in the predictive performance for each case.

For the **shallow track** in Figure 3.37, stable learning is accomplished after twelve learning experiences ($LE = 12$) and a retinal field with an average value of $TOTAL_{RF} = 0.6$ is developed. The average track deviation is $\psi = 5.0571$ and the predictive performance is resembles the retinal field to some extent. The predictive performance has a maximum of $0.8 < 1.0$ lower than the maximum value of 1.0 which would indicate the complete elimination of the

Table 3.14: Table of the reflex input entropy used for the calculation of the Predictive Performance in Figs.3.37,3.38,3.39. t_L is the time until the weights have stabilised. The distance between the reflex pixels and the predictor was always set to $y = 6$.

track type	y	$\underbrace{H(E_0)}_{t=0}$	$\underbrace{H(E_0)}_{t=t_L}$	t_L
shallow	6	0.149802	0.0	1000
middle	6	0.152688	0.0	1250
step	6	0.154852	0.0	2500

reflex pathway and total controllability of the robot's actions. In this case the weights (Figure 3.37(C)) and the Predictive Performance (Figure 3.37(D)) are nearly identical. The total synaptic weights (Figure 3.37(B)) is lower compared to the middle and step track because the robot manage the track without any particular effort. The shape of both the receptive field and of the Predictive Performance show that the robot is using essentially a triangular group of pixels on the top left of the retina. The highest weights are located close to the reflex and thus will generate stable correlations between predictor and reflex. The robot does not attempt to use the pixels on the top of the retina which does not generate reliable correlations.

For the **middle track** in Figure 3.38, stable learning is accomplished after nineteen learning experiences ($LE = 19$) and a retinal field with an average value of $TOTAL_{RF} = 1.4$ is developed. The average track deviation is $\psi = 3.8572$ and the predictive performance has a maximum of 0.91. In terms of weights the robot is using a diagonal and straight group of pixels. Comparing the weights (Figure 3.38(C)) and the Predictive Performance (Figure 3.38(D)), it is evident that while the weights are high along the diagonal line, pixels further up are not strongly utilised as pixels closer to the robot. This is due to the already very steep angles which can no longer be used to generate a smooth steering action. One could say that the robot is taking a higher risk by increasing the weights further out but they do not contribute as strongly to the Predictive Performance as the ones closer to the robot. However, overall the robot still manages this track with ease which is reflected in the high Predictive Performance values and low track deviations.

For the **step track** in Figure 3.39, stable learning is accomplished after eighteen learning experiences ($LE = 18$) however the robot is not able to stay on the track without learning. A retinal field with an average value of $TOTAL_{RF} = 1.5$ is developed. The average track deviation is $\psi = 6.5$ and the predictive performance is different from the retinal field. The retinal flow before learning does not have any particular structure as the robot is not able to stay in track with the only reflex. The predictive performance has a maximum of 0.45 and indicates that the robot is using few pixels in the lower left corner. Here, I have greatest difference between weight distribution (Figure 3.39(C)) and Predictive Performance (Figure 3.39(D)). While the weights are higher further away from the symmetry axis, the Predictive Performance is only high close to the robots reflex sensors indicating that the pixels further out are not able to improve the robots behaviour. The last example demonstrates that there is a distinct difference between Predictive Performance and weight distribution. It clearly shows that high weights are no guarantee for success in terms of closed loop performance. Instead, the Predictive Performance gives us a much better indication of whether a certain pixel contributes

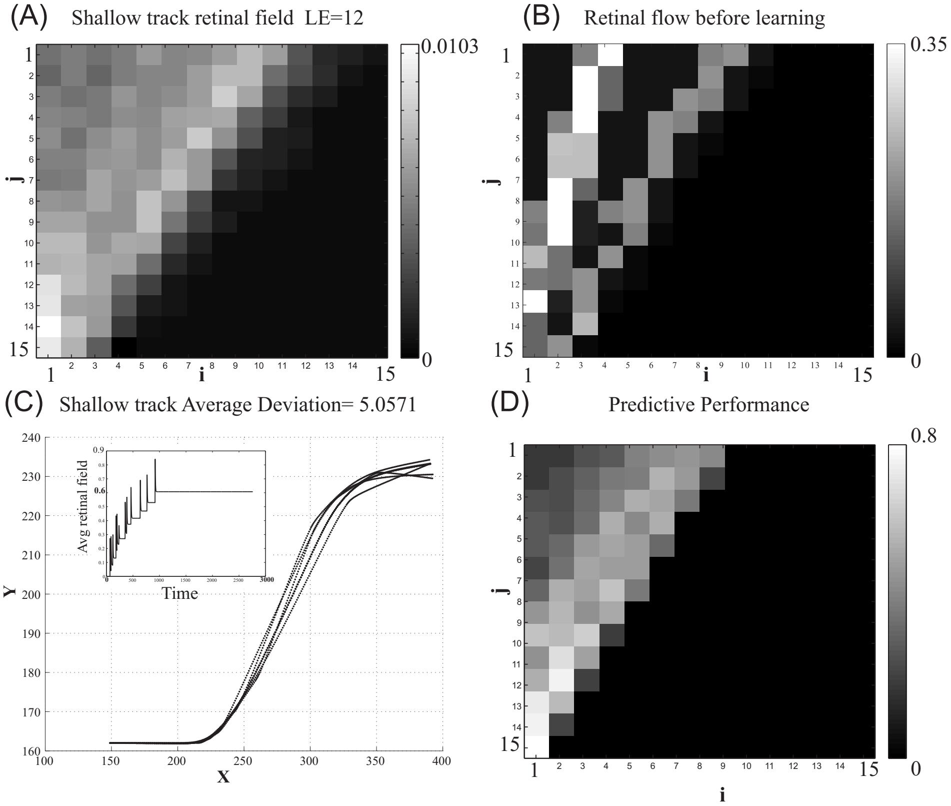


Figure 3.37: **A)** Retinal field developed during learning in the shallow track configuration as in Figure 3.35(A). The learning experiences required to have stable learning were 12. **B)** Retinal flow before learning $\mu = 0$ achieves a maximum of 0.35 bits. **C)** Trajectories produced by the robot during learning. The controller stabilizes its average retinal weight at about $TOTAL_{RF} = 0.6$ after 1000 time steps. The average deviation of the track is $\psi \simeq 5.05$ **D)** Predictive Performance for every pixel. The robot is using mainly a diagonal strip with a peak in the left bottom corner.

to the closed loop performance as shown by the simulations.

Predictive Performance during learning So far I have calculated the Predictive Performance at the end of learning. However, while the robot traverses along the line it will learn along the worst experience and then only continue to adjust its weights when it encounters a more challenging turn. Fig. 3.40(A) shows the maze track experiment where the robot has to master a track where it encounters different levels of difficulties. When the robot is learning on the maze track as in Figure 3.41(B) there are 2 stages of learning:

- the first one happens from the beginning to point P1.

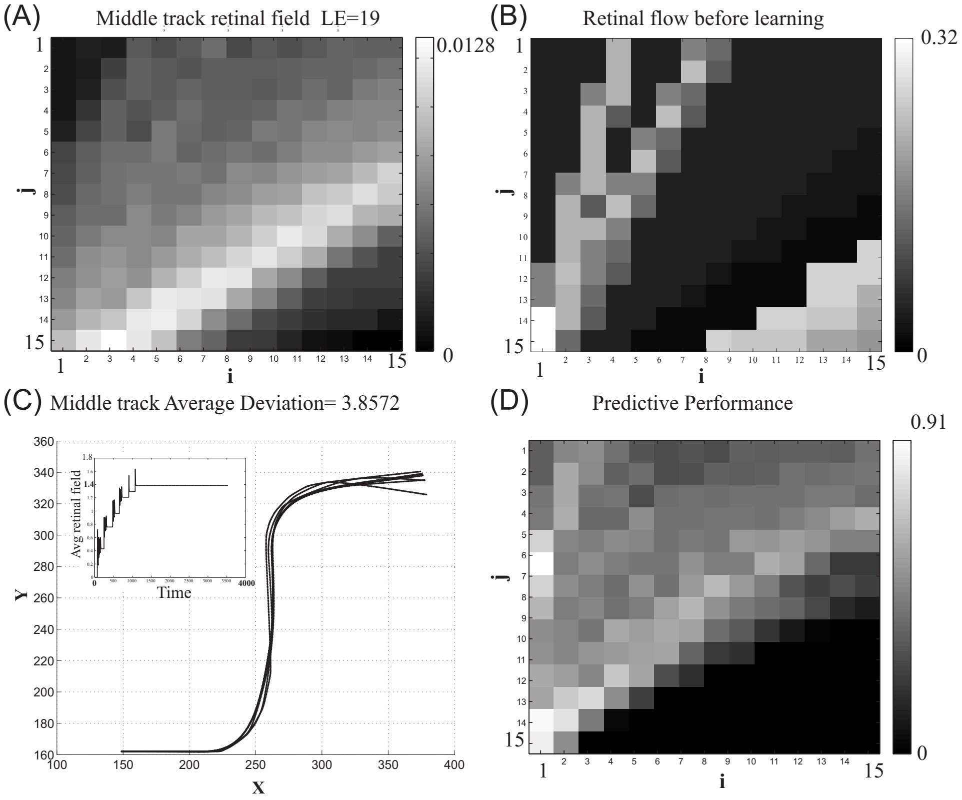


Figure 3.38: **A)** Retinal field developed during learning in the shallow track configuration as in Figure 3.35(A). The learning experiences required to have stable learning were 19. **B)** Retinal flow before learning $\mu = 0$ achieves a maximum of 0.35 bits. **C)** Trajectories produced by the robot during learning. The controller stabilizes its average retinal weight at about $TOTAL_{RF} = 1.6$ after 1250 time steps. The average deviation of the track is $\psi \simeq 3.85$. **D)** Predictive Performance for every pixel shows that the robot is using a wider area compared to the shallow case and with higher maximum of 0.9.

- the second one happens from point P2 to point P3.

Therefore the predictive performance and retinal field is computed for the 2 stages in 3.41(C,D,E,F). Remember that learning is error driven and it is only triggered when the reflex is utilised. The robot manages the track with and without learning, however learning reduces the track deviation from $\psi = 5.20$ before learning (Fig. 3.40) to $\psi = 3.86$ after learning (Fig. 3.41). In Fig. 3.40(A) the trajectory is approximated by a broken line whereas in Fig. 3.41(A) the trajectory overlaps almost perfectly to the track thus indicating a high performance. The values required to compute the Predictive Performance for the reflex case are reported in Table 3.15.

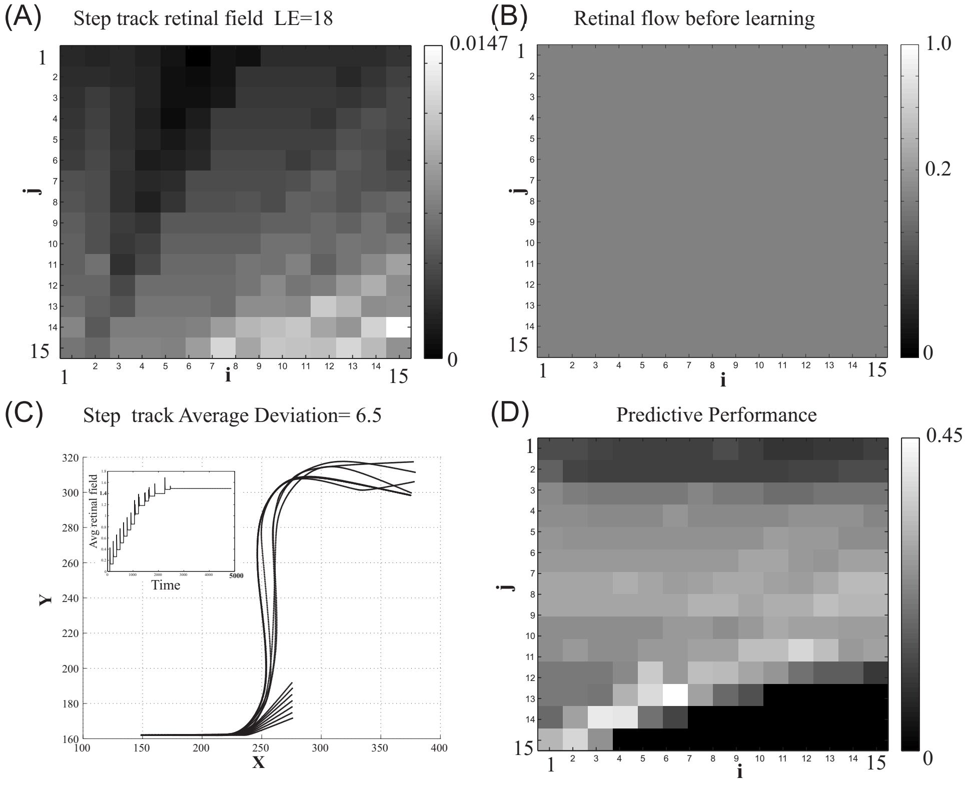


Figure 3.39: **A)** Retinal field developed during learning in the step track configuration as in Figure 3.35(A). The learning experiences required to have stable learning were 18. **B)** Retinal flow before learning $\mu = 0$ stays at about 0.1 bits. **C)** Trajectories produced by the robot during learning. The controller stabilizes its average retinal weight at about $TOTAL_{RF} = 1.44$ after 2600 time steps. The average deviation of the track is $\psi \simeq 6.5$. **D)** Predictive Performance for every pixel. The robot is using a wider area but with lower values.

Fig. 3.41(A) shows how the mutual information of the retina looks like during a purely reflex behaviour ($\mu = 0$) and is necessary to compute the Predictive Performance for the latter case Fig. 3.41(B) when learning is enabled $\mu > 0$. Learning is happening whenever the reflex is triggered. From the start until P1 the robot learns continuously because the reflex is used heavily and the weights grow (see the total receptive weight value in Fig. 3.41(B)).

Then at P1 the receptive field controls learning without resorting to the reflex and the weights stabilise. This works fine until the robot drives into a very steep curve at P2 where the reflex had to be used again and learning kicks in until the robot reaches more shallower curves at P3. In order to calculate the Predictive Performance at intermediate points I need to average over a certain period of time. Looking at the total weight development (Fig. 3.41(B)), I can therefore calculate the Predictive Performance for two sections defined as Stage 1 and Stage 2.

track type	y	$\underbrace{H(E_0)}_{t=0}$	$\underbrace{H(E_0)}_{t=t_L}$
maze stage 1	6	0.130942	0.0
maze stage 2	6	0.130971	0.0

Table 3.15: Table containing the mutual information values for the reflex in the maze track before and after learning.

The Predictive Performance increases from a maximum of 0.44 in stage one (Fig. 3.41(D)) to a maximum of 0.84 in stage two (Fig. 3.41(F)) thus indicating a relevant step in learning and the level of difficulty. During the first stage the Predictive Performance shares some similarity with the weights (Fig. 3.41(C)) but in stage 2 there is a strong difference (Fig. 3.41(C,F)) because the Predictive Performance is high in the upper left triangle of the receptive field. This again demonstrates the added value of calculating the Predictive Performance to assess which pixels of the retina are actually contributing to the success of the robot and which do not.

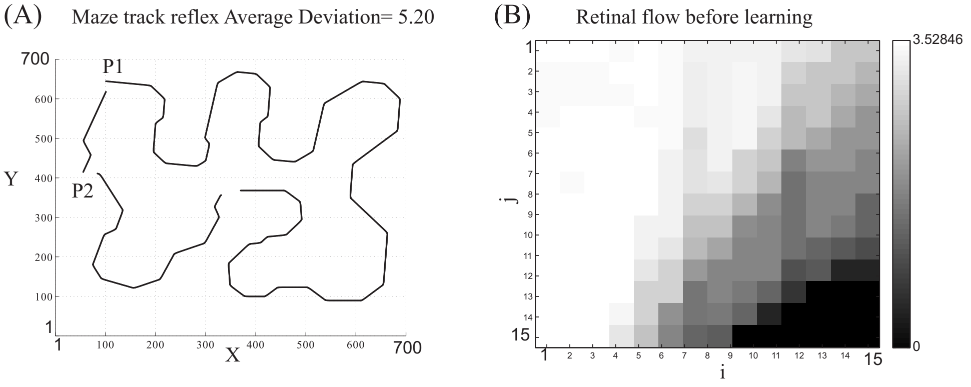


Figure 3.40: **A)** Trajectory of the robot in the maze track when only the reflex is used ($\mu = 0$). **B)** Mutual information $MI(Z, E_{1,i,j})_{t=0}$ for the retina computed during a track run

3.5.7 Results: application of PP to social systems

The predictive performance can be applied then to the social system setting described in Section 3.3 where I already computed the required values. Figure 3.42, left column shows the outcome of computing the predictive performance measure for the food and others attraction behaviour. The white bars contain the PP measure for the attraction behaviour PP_{Fo}, whereas the black bars contain the PP measure for the others attraction behaviour PP_{Af}. I have applied the measure for 3 different scenarios:

- a group of $N = 10$ agents and $M = 5$ food sources, generates five seekers and five parasites. The agents identified with $1, 7, 3, 9, 2$ have $PP_{Fo} > PP_{Af}$ therefore they are

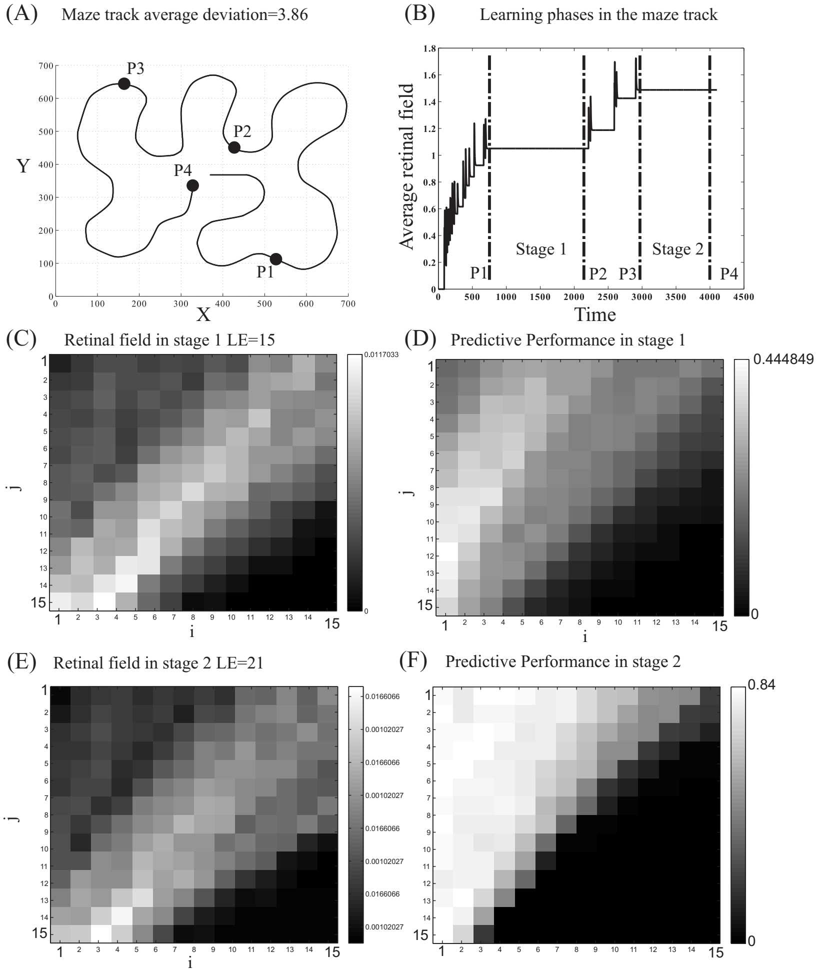


Figure 3.41: **A)** Trajectory of the robot in the maze track during a full learning session. **B)** The average retinal field plotted against time. The learning is stable in Stage 1 and Stage 2. There are 2 phases during the run marked with $[P1, P2]$ and $[P3, P4]$ where the robot does not learn anything new. **C)** Retinal field is constant during Stage 1. **D)** Predictive performance in Stage 1. **E)** Retinal field is constant during Stage 2. **F)** Predictive performance in Stage 2.

seekers. The agents identified with $5, 8, 4, 10, 6$ have $PP_{Fo} < PP_{Af}$ therefore they are parasites.

- a group of $N = 10$ agents and $M = 2$ food sources, generates two seekers and eight parasites. The agents identified with $6, 5$ have $PP_{Fo} > PP_{Af}$ therefore they are seekers. The agents identified with $3, 9, 2, 7, 8, 4, 10, 1$ have $PP_{Fo} < PP_{Af}$ therefore they are parasites.

- a group of $N = 10$ agents and $M = 8$ food sources, generates eight seekers and two parasites. The agents identified with $5, 9$ have $PP_{Fo} > PP_{Af}$ therefore they are seekers. The agents identified with $4, 10, 3, 1, 2, 7, 8, 6$ have $PP_{Fo} < PP_{Af}$ therefore they are parasites.

This outcome is compared with the weight development as shown in Fig.3.42, right column where the weight level for each agent is shown. There is a critical observation between the discrepancy of weight levels and predictive performance: high weights does not necessary mean high performance. For each case one can notice that:

- a group of $N = 10$ agents and $M = 5$ food sources (see Fig. 3.42(a)): agents numbered 7 and 3 have an equal performance as agents 9,2 $PP_{7,3}(Af) \simeq PP_{9,2}(Af)$ although their weights are lower $W_{7,3}(Af) < W_{9,2}(Af)$.

- a group of $N = 10$ agents and $M = 2$ food sources (see Fig. 3.42(b)): agents numbered 6 and 5 have a different performance $PP_5(Af) > PP_6(Af)$ although their weights are similar $W_5(Af) \simeq W_5(Af)$. Agent 2 has $W_2(Fo) \gg W_2(Af)$ but its predictive performance is similar $PP_2(Af) \simeq PP_2(Fo)$

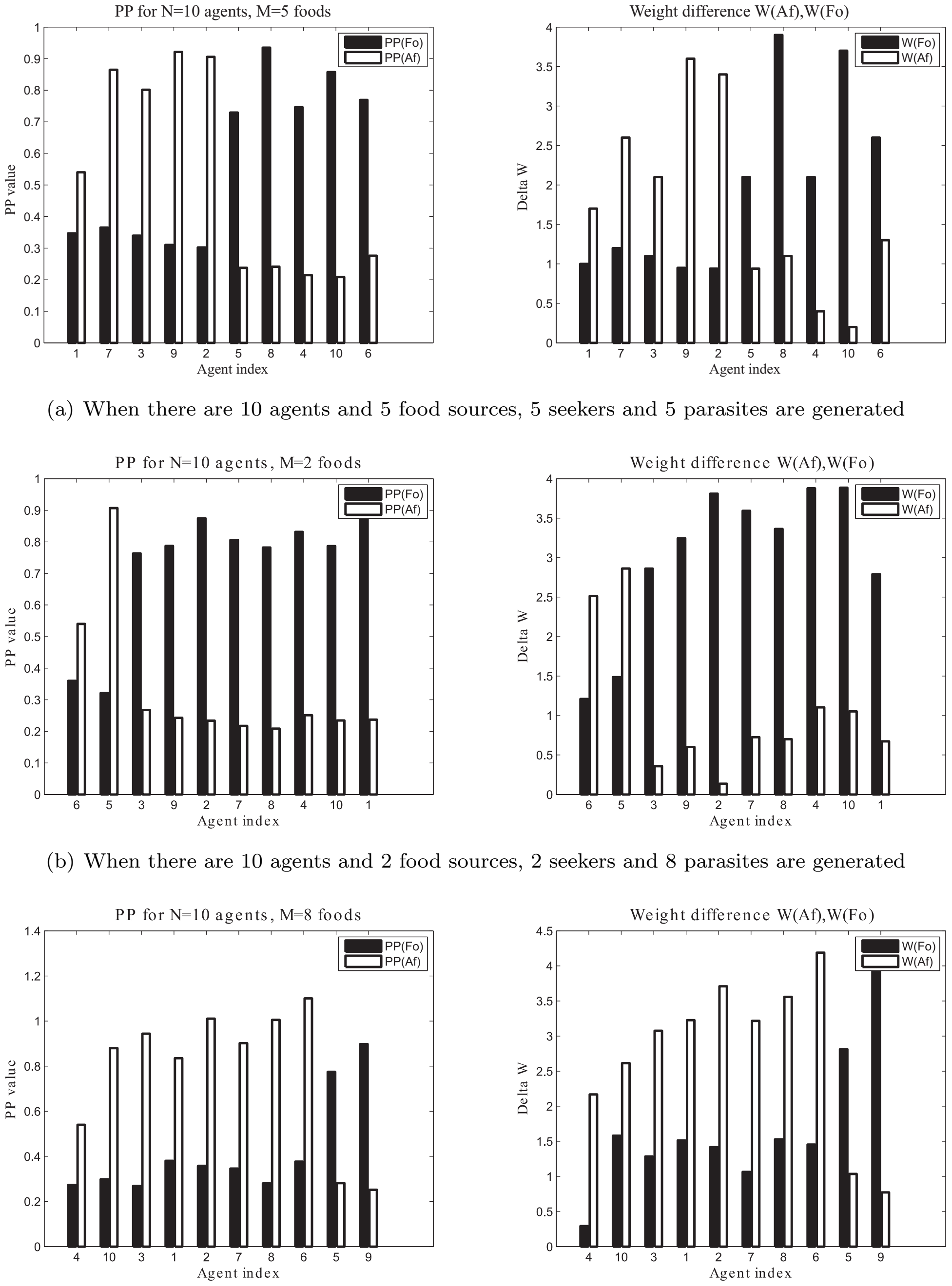
- a group of $N = 10$ agents and $M = 8$ food sources (see Fig. 3.42(c)): agents numbered 5 and 9 have an equal performance $PP_{5,9}(Af) \simeq PP_{5,9}(Fo)$ although their weights are differently distributed $W_9(Fo) \gg W_9(Af)$.

Again like the previous retinal field case, the weights are not a reliable measure of the performance of the agents. Especially in a highly dynamical social system, agents can be lucky and just find food by chance. One could be tempted to correlate the $PP(Fo), PP(Af)$ with the number of successful food bites or food stolen from other agents but this has the same limits of looking at the weight. A corresponding verification via an objective measure like ψ in Eq. 3.140 would require a complex trajectory analysis for each agent in relationship to each others' trajectories and therefore was not attempted for computational issues and lack of time. However basing my assumptions on the previous track follower, I am confident that the predictive measure can be trust and so agents $5, 1$ are the best performing agents in Fig. 3.42(b).

According to this new results, I can state that the weight are an actual representation of the social division. However in the future if the developer wants to use more complex agents in artificial societies, it will be necessary to compute the PP for each behaviour and then compare the different behaviours rather then relying exclusively on the weight development.

3.5.8 Discussion

The original Shannon's Information Theory (Shannon and Weaver, 1949) has been applied to closed loop systems in several studies like Ashby (1956); Tishby et al. (1999); Touchette and Lloyd (2000a). The scope of my predictive performance measure is to unify the Ashby's original theory of requisite variety Ashby (1956) with the recent frameworks based on mutual



(a) When there are 10 agents and 5 food sources, 5 seekers and 5 parasites are generated

(b) When there are 10 agents and 2 food sources, 2 seekers and 8 parasites are generated

(c) When there are 10 agents and 8 food sources, 8 seekers and 2 parasites are generated

Figure 3.42: Predictive Measure (PP) computed in the social system for: **(A)** 10 agents and 5 food sources, **(B)** 10 agents and 2 food sources, **(C)** 10 agents and 8 food sources. Left column contains the PP measures with white bars for the food attraction behaviour $PP(Fo)$, black bars for the other's attraction behaviour $PP(Af)$. Right column contains the weight developed after the system is stabilised for the same behaviours $W(Fo), W(Af)$.

information (Tishby et al., 1999) and Bayesian models of perception-action loop Klyubin et al. (2004, 2005, 2007, 2008). The Information Bottleneck (Tishby et al., 1999) is an information theoretic framework that finds concise representations for an 'input' random variable that are as relevant as possible for an 'output' random variable. This framework has been used successfully in various supervised and unsupervised applications but cannot be used as a measure of performance in closed loop controllers. Building on this initial theoretical work, additional studies were done on closed loop-systems from an agent-perspective considering the information processing properties of such system in the context of what would be beneficial for the agent itself (Klyubin et al., 2007, 2008; Prokopenko et al., 2006; Lungarella et al., 2005; Lungarella and Sporns, 2006). An interesting agent-centric measure called "empowerment" was introduced by Klyubin et al. (2005, 2008). Empowerment is defined as the maximum amount of information that an agent could send from its actuators to its sensors via the environment, reducing in the simplest case to the external information channel capacity of the channel from the actuators to the sensors of the agent. The empowerment is then used as an utility function that can be maximised by the agent's behaviour or by genetic evolution to produce meaningful states. The empowerment can also be measured to assess the performance of a general adaptive agent but it is necessary to disregard the actual behaviour of the agent and to model how the agent could behave in principle (disregarding the actual behaviour of the agent can be imagined as removing the agent's controller and studying the remaining empty shell which is the agent's body).

The fundamental difference between the Predictive Performance and empowerment is that the first is used to quantify the performance of a general adaptive controller whereas the second is used to drive the controller and at the same time to measure optimality.

Some other studies are focusing specifically on adaptive closed loop systems (Porr et al., 2006; Tomas et al., 2010; Lungarella and Sporns, 2006). Lungarella et al. (2005) has proven that for a saliency based attentional behaviour -based on a PT camera- the spatial mutual information increases and entropy decreases around the foveation point. The information measure was only computed for the visual input of the PT camera whereas my predictive performance measure takes into account the interplay of the sensory motor loop for the driving robot. Another essential difference with this study is the use of a purely reflex behaviour whereas my approach includes both reflex and learning behaviour.

A similar approach was used by Der et al. (2008); Ay et al. (2008) where the authors defined a predictive information measure PI as the mutual information between past and future sensor values to estimate the adaptation of a mobile robot to its environment. The mobile robot used in their study was a purely reflexive controller described by a parameter c which was chosen to simulate different behavioural regimes and how the PI was changing. Again this study is similar to Lungarella et al. (2005) because is based on a reflex controller and considers only the input: the transfer entropy is computed between the visual input S and the motor outputs M for different robot implementations with and without learning. In the experiment, when the robot is using reward-based learning, the transfer entropy is able to track the attention from red objects to blue objects following a change in the reward signal. This result is consistent with my observations on the visual flow which shows how the mobile robot has been adapted to different track shapes. My Predictive Performance goes further than just measuring the information flow because is able to identify if the agent is learning and being a scalar value avoids the complexities associated with multi dimensional analysis.

Tomas et al. (2010) makes a very complete study about the adaptive properties of driving robots in a square and circular environments. The mobile robots are using ICO learning but have only 2 spring antennas rather then a retinal input like my experiment. The author is able to predict the temporal development of the weights by using an analytical model which takes into account the predictive and reflex timing of the input events. The author then measures energy, input/output ratio and output entropy to estimate what is the best antenna ratio for a given environment. Although their modelling approach is correct, to compute the input/output ratio one must be able to separate the output contribution of the reflex from the predictor. This is not possible in a black box scenario whereas the observer is not able to discriminate the contribution of the reflex and predictor input. The speed of learning is then computed as the maximum value of the input/output ratio and together with the path entropy is used to measure the optimality of the robot. The path entropy is essentially the output motor entropy and thus indicates the complexity of the trajectories generated. The main difference with my work is that I summarize the performance of the agent to one value which tells us how good the agent is learning.

The development of the predictive performance is motivated by the original paper of Porr et al. (2006) where the predictive information is computed by summing the weights of the ICO learning rule (Porr and Wörgötter, 2003): the higher the weights the higher is the predictive information. There are 2 issues with this approach: firstly I trust what the agent has learned without looking at the environment's feedback, secondly it can only be applied to ICO/ISO learning agents. In the simple case of Porr et al. (2006) where the robot is using only one ICO neuron the predictive information is reliable but it cannot be applied to my case where the retinal input is multi dimensional.

The development of visual receptive fields, for example in the primary visual cortex, has been an intriguing problem addressed in numerous studies (Olshausen and Field, 1996; Blais et al., 1998; Weber and Obermayer, 1999; Hurri and Hyvärinen, 2003; Krding et al., 2004; Wyss et al., 2006). Whereas in these previous studies the agents operate in open-loop, my agent learns in a closed loop manner as proposed by McKinstry et al. (2006). The development of the receptive field has been investigated already by Kulvicius et al. (2007, 2010) which shows that RF can drive the motors of the robot in order to create better and more stable behaviour, and that development will stop as soon as the system has obtained behavioural stability after learning.

In summary, in this study I have analysed an adaptive predictive controller with the intention to quantify the information used effectively by the robot before and after learning. I introduced the predictive performance to measure the learning ability of the robot in different tracks. The robot is facing tracks with increasing difficulty -increasing curvature ratio- and is always learning to follow the tracks. I demonstrated that the predictive performance computed from the agent's perspective is consistent with the objective performance on the track. Additionally I argue that there is a limit in the potential information that an agent can learn, that enable us to set an upper bound for normalisation purposes. So the predictive performance help us to measure how much information flow the robot is using to achieve its goal without being biased by the absolute weight development. The predictive performance can be applied to every type of predictive adaptive control as long as the predictive and reflex inputs can be identified and is a useful tool that can can be used to give an objective estimate of the performance by evaluating information at the subjective level. The predictive performance is also very useful to determine the performance in the social system scenario where the highly dynamical setup does not allow the estimation of an agent performance by looking at the weights.

Chapter 4

Conclusion

4.1 Summary of results

This thesis has developed a computational model for the implementation of artificial societies based on the theoretic foundation of Luhmann. The societies are based on software agents that learn simple avoidance and attraction behaviours by means of a biologically inspired Hebbian rule. The intra agent communication is based on a minimalist implementation of Luhmann's communication model and is simple enough to generate the self-organising behaviour of social sub-division. The social division was assessed initially by looking at the synaptic weight development of each agent individually. This approach implies that the agents are considered as white or grey boxes, which means that the approach is not feasible if I consider a general agent whose internal status is not accessible (black box). Therefore I have then firstly developed two input measures called **maxcorr** and **AI**, secondly input/output measures **MI** to reflect the complexity reduction in their agent's behaviour and a single called **Predictive Performance** to gauge the learning performance of the agent. The measure has proven successful in measuring not only the behavioural reduction in a social setting but also the learning performance of the agent. This measure is general enough to be applied also to other learning controllers, for instance a Q-learning avoiding robot. The strength of this approach compared to previous work is that it is an information based measure that can be applied to real agents as well as simulated agents. Previous models in literature are based on discretised models or strategic games. In the following section 4.1.1, there is a comparison of previous work done in terms of information measures. In section 4.2 there is a description of possible extensions of the social system model, a potential application and a better analysis based on model checking. In section 4.3, I introduce some interesting topics as well as philosophical questions about the relationship between information theory, the theory of mind, psychology, language and neuroscience. In section 4.4 there is a small overview of the existing commercial systems relying on social robotic systems.

4.1.1 Discussion

The most relevant work that was done in the past about the implementation of Luhmann's principle in a computational model is the one of Dittrich et al. (2003). The model is implemented as a language game where agents are learning during mutual interactions. The social model that was developed in this thesis, although it does not include a complete communication protocol described by Luhmann and used by Dittrich, it is capable of generating sub-systems. It also has mainly 2 advantages:

- it is a real time simulator where the agents interact and learn continuously from each other rather then being limited to a simulated game.
- it can be implemented on a real robotic system as described in the Appendix

The other advantage, compared to Dittrich et al. (2003), is that the agents integrate action and communication in a very transparent manner thus facilitating a future expansion of the system. The communication used in the model is mono directional and one to many: this has an operative advantage in terms of fast response times if the system has to adapt to environmental changes.

As a comparative analysis, previous works in information measures was performed by the following authors, that used information theoretic cost functions to optimise the agent's behaviour:

- Polani et al. (2004) evolves controllers to maximise the information transfer of the sensory-motor loop (empowerment) and discovers that to use memory efficiently they perform compression as in Figure 2.18.
- Pfeifer et al. (2008) uses mutual information to generate information structures by motor feedback.
- Ay et al. (2008) maximises the excess entropy (the mutual information between past and present) of the agent's input thereby changing the controller's parameter to achieve a working regime (exploratory and sensitive to the environment) for the robot.

There is a substantial difference between the afore mentioned approaches and the one used in this thesis: the predictive performance discussed in Section 3.5 calculates the learning ability of a general adaptive controller based on the information flow. This is different from the other study by Polani et al. (2004); Lungarella et al. (2005); Ay et al. (2008) which use the information flow as a reward signal for the agent to learn. Nevertheless there are compatible results that shows how the two approaches are complementary. For example the experiment made by Lungarella et al. (2005) where he computed entropy measures on a saliency-driven attention task, where a camera foveates red blocks. The entropy for the foveation case is less than the random case: this means that a closed loop system induces statistical regularities in the information flow. My results coming from the social system application yields a similar concept: agents regularise their inputs by selecting information which affects their motor behaviour. There is a mutual relation between perception, information and action: agents select the information which in turns change their behaviour and their predictability. Indeed in Fig.3.25(D) the system is unstable when every agent is using all the information and thus producing non predictive behaviour. But when agents start to select the relevant information, they simplify their behaviour and intrinsically reinforce the stability of the system since agents mutually benefit from the increased predictability. Therefore the Predictive Performance is also used to measure the degree of behavioural selection of each agent during the social division as also described in Section 3.5.7.

Another interesting experimental work in the field of performance measure was produced by Kolodziejski and Kulvicius (2009) where the measure was the argument of the maximum cross correlation between the antenna's events $\bar{\tau}$ in a fixed time window. Kolodziejski and Kulvicius (2009) show that for more complex environments $\vec{\tau}$ has a larger deviation compared to the case of more simple environments.

The predictive performance can also be applied to reinforcement learning (Sutton and Barto, 1998) as demonstrated in Section 3.4 where the information flow was computed for a simple robot avoiding obstacles. Although the author did not have time to compute the predictive performance, only the information flow, there are no evident limits that will stop the computation of the predictive performance.

The predictive performance was finally applied to the social system scenario as demonstrated in Section 3.5.7 to show how agents select either behaviour in terms of information flow. This analysis show how the agents are selecting the information path and how is related to their weights' development. It also shows that performance cannot be based on the analysis of the weights but can only be reliably assessed via the predictive performance.

An intuitive and potential extension of the predictive performance is to classify the type of learning exploration or exploitation. So far I have only verified that the PP measure is normalised between 0 and 1, because in the second term I assumed that the information reflex flow before learning can be transferred totally to the information predictive flow after learning. However theoretically an organism which explores the environment could discover new relationships from the environment thus increasing the information flow after learning. In this case one could say that if $0 \leq PP \leq 1$ then the agent is only exploiting the environment, whereas if $PP \geq 1$ then the organism is exploring the environment. I was expecting from the Q-learning experiment to achieve something similar, but due to simplicity of the task I did not observe $PP \geq 1$. A better designed task could prove this property and can certainly a reason for a future study.

The Appendix 5.3 contains a more elaborate analysis of the weights' development during the sub-system formation by using a clustering analysis in the phase space of the weights. This sort of analysis provide a better qualitative approach to the determination of the sub-system separation.

The following section describes what modelling choices were done during the research and how they were justified.

4.1.2 Modelling choices

In the theory of social communication Luhmann (1995) and previous cybernetic experts asserted that an artificial agent or organism have mutual expectations towards other agents. An example which explains this condition is the difference interaction between a person and an object or a person with another person. We have a direct expectation for an object because we know that it only adheres to the laws of gravity and we know that throwing it will make a parabolic trajectory in the air. We have a mutual expectation between "ego" me and "alter" you, because we both try to predict what we are thinking. In a way that is more similar to a chess game where each player tries to predict the next move of the other in order to maximize his chance of victory. In the deterministic memory based model developed by Dittrich et al. (2003), social order arose from the social interaction of agents. Therefore my choice in the model was to separate the prediction of the world/environment from the prediction of other agent's actions. The identification of alter and ego was not included in the model and is left for future work, however most advanced organisms like mammals and primates are able to recognize their interlocutor and so maintain different expectations according to their previous interactions. The language used to communicate the food information is similar to a sign language which is based on the action layer, the most simple example could be the everyday traffic flow of cars. The left light arrow indicates that the car in front of us is going to turn left and vice-versa. This kind of sign language is used in the animal world and has been extensively studied by Smith and Harper (2003). A good example is the evolution of the ritualisation of the mating and fighting behaviour as briefly described in Figure 4.1.

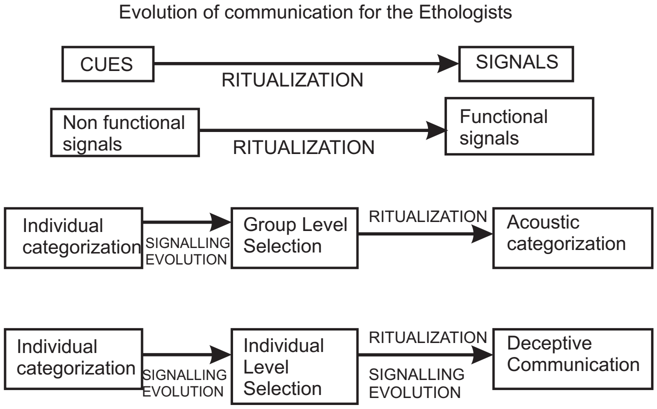


Figure 4.1: This diagram shows how Ethologist explain the evolution of animal signalling or language

A more advanced communication language uses a symbolic language which is based on top of the action or sign language: primates developed a more advanced mean of communication using sound and developing specialized areas of the brain like the “Brocha area” in humans and equivalent structures in the singing birds (Lai et al., 2001; MacDermot et al., 2005).

A sign or ritualised language is constrained by the environment and cannot develop further, this is why a better model would require the use of a symbolic language which I am going to describe in the next section.

4.1.3 A theory of language

This section contains first an introduction to the theories regarding the development of human and animal languages. It contains a brief summary of the current knowledge about the biological roots of language formation and experiments which try to replicate artificial language.

The communication model used in my artificial social system is very basic and mimics essentially the simple mechanism of signals used in simple animals like primates or event insects. The power of a more abstract or symbolic language has been assessed in lesion brain studies where basically it was discovered that intelligence is rooted in language. The recent discovery of the FOXP2 gene -dubbed the ”language gene”- has provided an astonishing example of the importance of the Broca's functional area; in humans, mutations of FOXP2 cause a severe speech and language disorder (Lai et al., 2001; MacDermot et al., 2005).

A model of language development, was achieved by Steels (1998, 1999), the talking heads experiment shows that a grounded language can indeed be evolved and contain many properties seen in natural languages like polysemy and synonyms. Steels (1999) has investigated how artificial agents can self-organize languages with natural-language like properties and how meaning can co-evolve with language. His hypothesis is that language is a complex adaptive system that emerges through adaptive interactions between agents and continues to evolve in order to remain adapted to the needs and capabilities of the agents. Thus a community of language capable agents can be viewed as a complex adaptive system which collectively solves the problem of developing a shared communication system. To achieve that, the community must reach an agreement on a set of forms (a sound system in the case of spoken language), a set of meanings (the conceptualisations of reality), and a set of form-meaning pairs (the lexicon and grammar). The experiments implemented interactive robots that were programmed to play language games and observed the characteristics of the languages that emerged; surprisingly the agents were able to self organize and develop a common language, which resembled many features of human like languages, without the help of an external teacher. The pre-requisite for the emergence of language is the cognitive and sensory-motor ability at the individual level because without the ad-hoc apparatus for exchanging information and the ability to categorize the environment it is impossible to develop a language.

The experiments shows that for a language to emerge there are several conditions:

- a common frame of attention
- the ability of perspective change
- a reliable system of communication
- adaptive learning

A sign language can emerge by interacting agents in the world: organisms can develop an alphabet of actions that will generate a predictive behaviour. Put simply, the agents have mutual expectancies from each other: if agent A sees a red square in front of him, it will produce a tone say at 150 Hz and agent B maybe will produce a tone say at 300 Hz. With time the agents will use a common alphabet to indicate different geometric shapes. This is possible because (A) both agents have the same or similar computational capacities and (B) both agents can imitate each others actions. Imitation is very important for the development of language and it has been discovered that this important function is implemented by mirror neurons both in primates and humans (Buccino et al., 2004). Even non primates like birds, must have a brain circuit generated by the gene PX64 to enable them to reproduce acoustic sounds of similar pitch. Birds that doesn't have this gene cannot develop a common language, even children who lack this gene are not able to develop a proper language. It turns out that if an agent is to be able to learn a language they must have a capacity to imitate and process in memory actions performed by his peers.

My current communication model on the contrary is limited or bounded to the properties of the environment: doing so limits the recursivity of a symbolic system. I need a higher process that is able to abstract the embodied actions' that happen in the world to a higher level of actions (let's call them symbols but they can be sounds or tones) that are able not only to refer to the objects of the real world but also to each other.

It is very easy to see the limit of an embodied language system: a well known studied effect is that one of the foraging domestic pigeons (which google sarcastically claims use a new page rankingsystem). Pigeons do have a good visual system but can detect food relatively well on the ground, once a pigeon finds a possible source of food it goes on the ground and randomly samples the ground to find the lucky spot! Another pigeon flying nearby will see his fellow pigeon wondering around the interesting spot and with some probability it will go on the ground and look for the food itself! If we repeat the step many times for each time a pigeon passes nearby we can see an entire storm of pigeons eating imaginary food! This kind of behaviour [reference] is a positive feedback mechanism which is based on a priory knowledge of the pigeon: the fact that if a fellow pigeon is looking for food on the ground then there should be something there! To some extent pigeons base their decisions on Bayesian inference. Surprisingly for very simple behaviours this also humans base their choices on Bayesian inference. This was shown in a very simple experiment that you can do when you feel bored: walk along the street with a friend and at a given time stop and look up at a point in the sky. After 1 hour you will get other 20 people doing that and so on. This primitive communication language happens if, as we said before, all the necessary conditions are met: a joint frame of attention for the pigeons looking for food on the ground and for the humans looking at some point in the sky, the ability to imitate actions (here we suppose to move to the same point in the space) and the ability to remember what the other are doing. How this a-priory knowledge has developed could depend on many factors, it's certainly due to the evolution mechanism as well as on our personal experience. For us it's very hard to get fooled by that trick more than one time as we may infer that when a group of people gather in that situation it has no importance, but for the pigeon it is another story. Pigeons will keep that behaviour because it is beneficial: on average they will be able to find some food and therefore will keep their a-priory estimate so that next time they will come back. For the unlucky pigeon that, by a series of unfortunate events, will not get any food, there will be a life of solitude and eventually an early retirement! The conclusion here is that language cannot be based solely on actions because, they are embodied in the environment where they are performed. Thus to develop a more complex language, the organism must be provided with an additional layer of sensory-motor loop designed or adapted for the purpose of communication.

4.1.4 Language model and control

A control model which includes language was used in Steels (1999) and is summarised in Figure 4.2 where two artificial agents play the role of the speaker and the hearer each time they meet each other in a rule based language game. The model adopted by Luc Steels includes similar features to Luhmann, the intentionality is achieved by the goal to reduce uncertainty, the utterance module also involves the information selection, and parsing the utterance is basically the understanding process.

In the talking head experiment (Steels, 1999), every communication round is composed of one speaker and one hearer. The speaker and hearer share a whiteboard (called "the context") full of geometric coloured shapes (triangles, squares, circles). The speaker, uses image segmentation to choose a topic like "the green triangle" or "the square in the top left corner". The speaker then, chooses a word from its dictionary to describe the topic, then it emits a "linguistic hint" to the hearer. Based on the linguistic hint, the hearer tries to guess what topic the speaker has chosen, and he communicates his choice to the speaker by pointing to

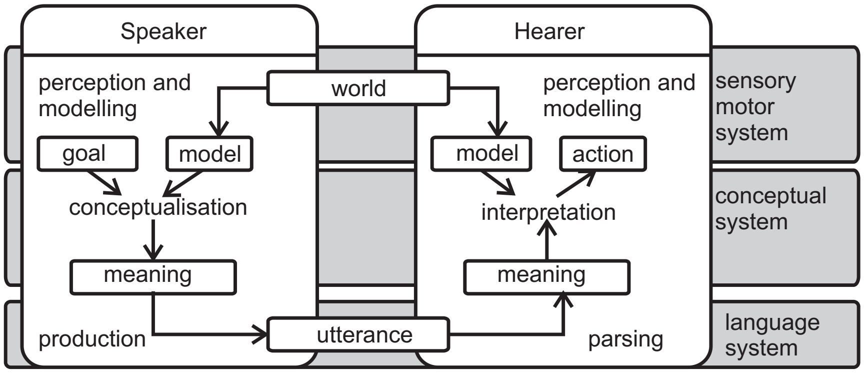


Figure 4.2: Both the speaker and the hearer have a layered architecture: the language system takes care of the production and parsing of the utterance, the conceptual system takes care of the understanding, the sensory-motor system includes the modelling of the world and the intention or goal of the agent.

the object. The hearer points by transmitting in which direction he is looking. The game is considered successful if the topic guessed by the hearer is equal to the topic chosen by the speaker. The game fails if the guess was wrong or if the speaker or the hearer failed at some earlier point in the game. In case of failure, the speaker indicates the topic he had in mind, and both agents try to "synchronize" their dictionaries to be more successful in future games. The talking head experiment shows that a grounded language can indeed be evolved and contains many properties seen in natural languages, such as polysemy and synonyms.

In my simulations what the agents are missing is the conceptual system layer of Figure 4.2 which generated concepts and extract meanings from the language system. The world model of my agents is quite simple because it basically reduces a multidimensional time series in a single integrated value which is the ICO weight. The world model is of course a bottle neck for the conceptual system but that does not imply that a simple conceptual layer cannot be developed. For example part of the plan in my simulations was to introduce such a conceptual layer ("telepathy" feature) whereby the agents shares their own weight with each other. When the agent receives the weight from one if his fellow, it can decide whether to use it or not. If the weight is beneficial, then is going to keep it otherwise is going to reject it. This simple mechanism allow agents to share their representation of the world and assigning a simple binary meaning: does it work or not? With this sort of communication, one would also expect a faster convergence in the self-organization property because agents does not have to experience everything but can just try to apply somebody's else knowledge. This feature unfortunately as explained in Figure 4.3 was not implemented for the lack of time but would be certainly improve the model complexity and possibly generate more complex collective behaviour.

It is also equally important to study animal language and see what are the main differences with the human language. Due to the complexity of the cognitive abilities of humans, in this Thesis I used models of language closer to animals rather then humans, and so I am going to describe in the following sections some experimental evidence of animal language.

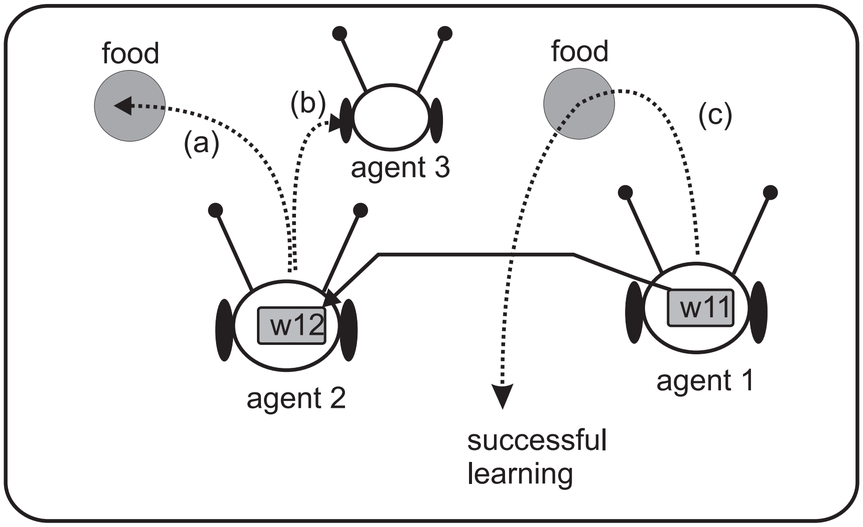


Figure 4.3: In this situation there are three agents. Agent 2 behaviour is to be attracted by agent 3 due to its current weight status $w12$. Agent 1 is going to learn the food attraction behaviour and consequently transmitting his weight $w11$ to Agent 2. Agent 2 will receive the communication from Agent 1 and will decide to try the new weight $w12$. If the operation is successful it will keep the new weight $w12$ and overwrite its previous one $w11$.

4.1.5 What is going on in an animal's head?

What do the signals given by primates, and their responses, tell us about how monkeys think? When we see an animal do something, it is tempting to assume that it's thinking as we would if we behaved in the same way. But, as we explain in the context of alarm calls, this need not to be so. Sometimes, however, by appropriate use of playback experiments, we can get answers.

4.1.6 Do signals convey information about the external world?

Some signals convey information about the signaller. For example the black and yellow stripes on a Cinnabar Moth caterpillar carry the message "I am distasteful', a fact about the signaller, not about the world external to the signaller. In our model, agents show to the other, their state of satedness: it's the same concept. But other signals do carry such information: for example, a bird alarm call carries the message "there is a predator close by" or in my framework, the agent signals the presence of a food resource in front of it by changing its colour. So what is the difference between the two cases? It's about what, if anything, goes on in the mind of the receiver of the signal. Is the receiver genetically programmed to react to the alarm (a pure reactive agent) or is the receiver formulating a hypothesis of the external world (a non reactive agent)? To be specific, when a Vervet Monkey hears a Leopard alarm, it climbs a tree. Does it do so because it forms an image of a Leopard in its mind and behaves accordingly, or because it follows the behavioural rule "when you hear that call, climb a tree" ? We know that a Vervet will behave appropriately when it hears a Leopard alarm, even when no Leopard is present. But what is going on in its head? Seyfarth and Cheney (2000) attempted to answer this question by habituating experiments. Summarizing their conclusion "Vervet Monkeys, therefore, appear to interpret their calls as sounds that represent, or denote, objects and events in the external world". Current imaging studies are shedding more light into the mechanism of humanoid brains but there is still a lot of unknown processes.

4.1.7 Do signallers intend to alter the behaviour of receivers?

Moller (1988) argues that subordinate birds gave false alarms to drive away more dominant individuals and thereby gain access to food. It requires only that individual birds learn that giving an alarm note increases their access to food: the calling bird does not have to think "if I give an alarm, other birds will think that there is a predator and fly away". An even more cautious interpretation is that the behaviour is not learnt at all: it's innate in all situation in which calling has been selectively favoured in the past. In my framework some agents cheat to decrease food competition, but to keep the model simple this behaviour is not learned, only a percentage of the population is cheating. In this example, there is no need to assume that an animal ascribes thoughts and beliefs to others. Humans certainly do. What of other primates? A summary of Dennett's classification of "intentionality" (Dennett, 1987):

- *Zero-order intentionality.* The signaller holds no beliefs or desires: a black and yellow caterpillar is a likely example.
- *First-order intentionality.* The signaller holds beliefs, but no beliefs about the beliefs of others. A Great Tit giving an alarm does not believe that there is a predator (if the signal is honest), or that the alarm will not increase its access to food (if the signal is a lie), but in neither case need it have any beliefs about what other Great Tits are thinking.

- *Second-order intentionality.* The signaller ascribes thoughts and beliefs to the receiver.

The existence of zero-order and first-order intentionality in animals should not be controversial. One problem is relevant for the second-order intentionality: the influence on the signaller of the presence of potential signal hearers. Vervets and others (including ground squirrels and chickens) do not call when alone. This shows that animals may be aware of the presence of other individuals before giving an alarm, but does not require that they ascribes thoughts to others. To summarise, although animals are influenced, when signalling, by the presence and relatedness of potential hearers, they do not seem to be influenced in their signals by the knowledge that hearers might be supposed to possess (e.g. a monkey already giving the alarm call is supposed to know that a leopard is present) Is there any evidence that signallers ascribe beliefs to others? A theory called "Machiavellian Intelligence" was formulated by Byrne and Whiten (1988). The specific thesis is that group-living primates have been selected to deceive other group members, and that this requires that they develop a "theory of mind": they are able to ascribe beliefs to others. There is a general agreement that primates do sometimes send signals which causes others to behave as if they have been deceived. However, as said before about the hawk alarm call, this does not require that the signaller ascribes beliefs to others: it is sufficient that the signaller learns by experience that the false signal has the desired effect on the receiver's behaviour. Human children, for example, develops the ability to ascribe beliefs when they are 4 years old (Wimmer and Perner, 1983). There is no doubt that some animal signals do potentially carry information about the external world, and that receivers of such signals respond in a way that would be appropriate if they had acquired that information. It is much harder to decide in particular cases whether the receiver in fact acquires the information, or whether it merely responds appropriately. According to the theory of animal signals (Smith and Harper, 2003), signals in my simulations can be regarded as:

1. the field G_{sated} in eq. 3.21 is an index signal, expressing a quality of the agent (its state of satedness) which cannot be faked.

2. the field G_{food} in eq. 3.29 could be regarded as a costly signal or a free signal

I did two different simulations considering, $P_e(t)$ as the efficacy cost needed to ensure that the information can be reliably perceived, $P_s(t)$ cost needed by the handicap principle (Zahavi, 1975) to ensure honesty. Efficacy cost is considered free in my simulation (see section 3.1.8):

1. G_{food} as an honest signal with $P_e(t) = 0$ and $P_s(t) > 0$ because competition for food will increase

2. G_{food} as a dishonest signal with $P_e(t) = 0$ and $P_s(t) = 0$ because the agent that hasn't food, does not have any disadvantage.

The model that I used is based on the zero order intentionality and one of the aim for future research is to achieve the second order intentionality.

4.2 Future work

An important improvement for the model will be to use the symbolic language module as described in section 4.1.3 with the double contingency feature described in section 2.3. With this kind of approach agents will show a motivated behaviour towards others, develop their own language and not just share a symbolic system (as proposed in the Parson's model). Thus one can have the power of a grounded language model and the potential for social interaction to build an accurate model of Luhmann's society. The *ICO*/*ISO* learning controller could still be used to implement the action layer, but other approaches will be required to implement the symbolic communication layer. There are also some other extensions and considerations that can be included in future models and are described in the following sections.

4.2.1 Model based checking for property verification

The research on agent-based learning systems currently relies on simulation results to infer the correctness of system properties. These inferences are derived from averaging set of simulation results. However an alternative approached based on model checking can be used to verify the properties of a learning system without the need of a simulation. In a preliminary study with a fellow PhD student Ryan Kirwan from the computer science department I have proved that it is indeed possible, with the correct abstraction model, to apply model checking to a dynamic learning agent and prove some properties for a multi agent non learning system and a single learning agent. The application of model checking to autonomous learning agents is novel and thus not straightforward.

4.2.2 Economic models of learning agents

Bayesian inference is a fundamental process behind human perception, memory and cognitive judgement. While Bayesian inference has been investigated at the individual level, there are few studies regarding the implications of using Bayesian decision making in a multi agent social scenario. Verschure (1998) argues that bayesian inference is an equivalent formulation to adaptive predictive control beacuse it is essentially a computational approach equivalent to the dynamical approach of neural based systems. Thus it is possible to use Bayesian inference in a decision making task which requires a selection between several actions i.e. an action policy. An interesting economic social experiment can be setup where the goal of the artificial agent is to win a virtual English auction. Every agent has a Bayesian predictive policy and/or an expectation about the others to decide the next move. Therefore the simulated model takes into account the mix of subjective and social knowledge. The social knowledge is based on the mutual expectation, a fundamental property of social systems. Luhmann hypothesised that high degree of behavioural dynamics can be achieved by using expectations as valuable knowledge for reducing contingency about each others' behaviour and goals. Therefore the author expects that a social based approach will generate a realistic dynamic behaviour even in auction based games. In the following sections the author describes a ABM system based on auction bidding that considers social expectations.

4.2.3 Homogeneity in societies

Luhmann did not pose any constraints on the homogeneity of social systems, because his social model is essentially "actor-free". This choice theoretical choice is quite good because it allows great flexibility to build heterogeneous societies. The only requirement is the need of a psychic system coupled to the communication system. For Luhmann a psychic system is a system able to generate thoughts, although there is a philosophical debate whether or not machines are able to think, dream or create, a psychic system can be implemented as a goal oriented behaviour and a language module as proposed by Luc Steel. It is useful therefore to distinguish between:

- homogeneous societies: composed of identical entities, either artificial agents or humans
- heterogeneous societies: composed of mixed entities like artificial agents and humans

There was a period of excitement in research after science fiction writers envisioned the integration of artificial intelligence entities in human societies. Note that the very first ”robots” in fiction, the neologism ”robots” from Karel Capek's R.U.R. (Čapek and Novack, 1920), were actually Artificial Humans and not the clanking metal humanoids we now associate with that term. A better and earlier term is Android from the greek andro- ”human” + eides ”form, shape” meaning an ”automaton resembling a human being”. The term was first mentioned by St. Albertus Magnus in 1270 and was popularized by the French writer Villiers in his 1886 novel L'È̀ve future. There were and there still are efforts in producing androids which can be accepted by humans, avoiding the famous “Uncanny Valley” introduced by Masahiro Mori as “Bukimi no Tani Gensho”. The latest androids are able to mimic the human aspect thanks to recent advantages in material structures (artificial hairs, silicon skin) The current problem is then to provide the androids with the intelligence to interact socially and safely with humans. Many researchers attempted the direct approach of designing social robots by looking at the single human-robot interaction with an engineering top-down approach. The design of social robots contains a variety of disciplines including: mechatronic, science of materials, psychology, neuroscience, haptic interfaces, voice recognition, speech synthesis, power systems etc. It is certainly an interesting field, but it is mainly driven by technologist and behavioural scientists,and thus is proceeding at a slow pace considering also the cost involved with building androids.

4.2.4 Symmetry breaking in collective decision-making

The self-organising property of social systems can be formulated in terms of decision making because effectively one can imagine that the agents has to make a collective choice about their division. Collective decision making in social systems is often driven by self-organising principles such as the choice of nest sites and food sources by ant colonies and the aggregation of bees (Franks et al., 2003; Beekman et al., 2009; Meyer et al., 2008; Kernbach et al., 2009). Similar dynamics are present in bacteria colonies (Reading and Sperandio, 2006) and even economic markets (Weisbuch and Stauffer, 2000).

In my computational model the agent (individual) needs to decide whether to obtain some food itself or steal the food from the others. Because each agent has no initial preference or bias, the agent needs to make a decision at the individual level based on his memory (synaptic weight) and actual sensory inputs (antennas). Symmetry breaking means that the society will reach a majority or unanimous decision. In my case that implies that there will be a non equal distribution of seekers and parasites. It may not be obvious why the social system must always operate under symmetry breaking even when there are 2 equally good sources. This in fact happens in nature where for example many species will converge on a single food source rather then equally distributing between the 2 equally good food sources (Camazine et al., 2001).

Symmetry breaking in self-organised decision making usually arises from the interaction between positive and negative feedback loops. The positive feedback in my model is the progressive weight increase of the synapse which orientates the agent toward one behaviour, for example food seeking. The negative feedback in my model is the collision resulting from a crowded group of agents going for the food. The balance between these 2 systems has been shown to be a stable and flexible enough decision system (Beekman et al., 2009; Meyer et al., 2008).

The most common analysis of this coupled system is via differential equations, but as stated before, the model is very complex to be described, especially because the agents are active learner and thus their properties change during time.

The alternative is to use a reduced statistical model which captures only the relevant property of the symmetry breaking. An early study in this field was done by Hamann et al. (2010a) who described the symmetry breaking in the honeybee behaviour and an emergent density classification task with a simple stochastic differential equation.

4.3 Information theory and control

4.3.1 On the perils of predictive learning

Predictive learning is not the best solution for every situation. Why? Because predictive learning is based on our subjective knowledge about the world statistic in where we live. To give a clear explanation about when predictive learning fails we can think in probabilistic terms. Suppose we have a black box that generates a stream of data, this can be the stock market, an auction on ebay or a football match. We don't know anything about the model behind the generation. It could be in the worst case a markov process that generates events with maximum entropy. Nevertheless in the short run we only observe a causal relation between event A and event B, predictive learning that in the general formulation finds the causal relationships between two events, A and B, and will infer that event B follows event A with probability 1! Predictive learning doesn't know the statistics behind the process generator because of its limited sampling capacities. If it was able to have an infinite sampling time (say the organism is immortal) it would experience all the possible pairings, and due to the ergodicity property of a markov process, it would infer that event A can be followed by event B with the same probability of being followed by event C. This sampling problem can cause what we define in daily life as hallucinations: a (conscious) perception in the absence of a stimulus. In the absence of a stimulus, our predictive ability is reduced to zero and so everything can be plausible and "real". An extension of predictive learning models has been introduced by Schmidhuber (2010):

> "What's interesting? Many interesting things are unexpected, but not all unexpected things are interesting or surprising. According to Schmidhuber's formal theory of surprise & novelty & interestingness & attention, curious agents are interested in learnable but yet unknown regularities, and get bored by both predictable and inherently unpredictable things. His active reinforcement learners translate mismatches between expectations and reality into curiosity rewards, or intrinsic rewards for curious, creative or exploring agents which like to observe or create truly surprising aspects of the world, in order to learn something new."

Schmidhuber rejects the original notion of the Boltzmann/Shannon surprise formulation from the early 1990s by posing two significant examples of uninteresting, unsurprising, boring data. A vision-based agent that always stays in the dark will experience an extremely compressible, soon totally predictable and unsurprising history of unchanging visual inputs. In front of a screen full of white noise conveying a lot of information, ”novelty” and ”surprise”, in the traditional sense of Boltzmann (1800s) and Shannon (1948), however, it will experience highly unpredictable and fundamentally incompressible data. In both cases the data gets boring quickly as it does not allow for learning new things or for further compression progress. Neither the arbitrary nor the fully predictable is truly novel or surprising/interesting - only data with still unknown but learnable statistical or algorithmic regularities are. This is a very good argument and it would be interesting to integrate the notion of maximisation of learning speed in future social models. For a more mathematical formalisation between entropy, learning and prediction, the Appendix in Sections 4.3.3,4.3.4.

4.3.2 Prediction or evolution

There are only two options to design artificial agents: either the designer can evolve reactive systems or adapt predictive systems to the environment. Evolution has luckily selected organisms which infer the causal structure of their environment to make predictions of their future actions or equivalently of the future stimuli. Most researchers will argue that I’m talking about different things, but a careful study of closed loop system can show that if the organism is able to predict what the next stimulus will be if he chooses an action, then he is also able to predict what the stimulus will be if he chooses not do anything. Imagine a cat observing a little mouse running in front of it, the cat will estimate the trajectory taken by the mouse at a given time and will decide if is worth trying to pounce on it or if the mouse is too fast and so waiting for a closer trajectory wastes less energy. This is a well known probabilistic dilemma: when the organism needs to be reactive and when the organism needs to learn?

The Shannon probability measure is a concave function of the distribution when expressed as $\sum_p p \cdot log_2(p)$: $H(p) = 0$ when $p = 0$ or $p = 1$. However the property is not valid if one only uses the logarithm as $\sum_p log_2(p)$ which is infinite at $p = 0$. So the best choice which maximizes the information is always somewhere in the middle. This poses another question: if an organism wants to have a maximally predictive state of the environment, why do anything since a stationary state produces less possible entropy! If this assumption was correct we would live in a stationary environment where organisms do very little as required by their survivor instinct. Well in the animal world there are uncommon animals which live in very boring environments like in the darkness of a deep ocean where their world is a flat surface with rare events happening without any clues. It turns out that the best organism is the one that is very fast to react to changes, prediction has little sense in this game. A star fish is the best choice, there is no need for huge brains with lot of computational power because it will be mainly wasted. Coming back to our land we can see how higher complex organism have developed different senses and very complicated brains to cope not with a complex environment but with a causal environment. The common mistake is to think that a complex environment requires a complex organism. This is not true! Even a fairly simple organism can generate a complex behaviour when placed in a complex environment. Ashby proved that even when the inputs are connected to the outputs with a simple proportional rule, the the variety will be transferred from input to output unchanged. A superficial observer will say that this organism has a rather complicated behaviour! Moreover if we feedback the motor action into the input according to a function f, coupling will generate an even more complex behaviour.

As already shown by Kolodziejski and Kulvicius (2009), output entropy was computed as the ration between the reflex output and the predictor output, allowing the author to separate the 2 contributions during learning. This approach of separating the different output types was necessary to avoid the afore mentioned problem of the variety transmission from inputs to outputs. The ICO/ISO learning, in terms of information theory, is an internal memory which integrates sensory information and to some extent compress the sensory information by discovering the causal relation between the predictor and the reflex. In this way then the total output entropy will not vary significantly and thus cannot be used as a parameter of learning as well as complexity.

4.3.3 Prediction and learning

The definition of predictive information: a quantity that measures how much our observations of the past can tell us about the future. The predictive information describes the world we are experiencing and has a direct link to its complexity. We as organisms collect sensory information in order to choose our actions (including our verbal communication) but we are only interested in the data that tells us something about the state of the world at the time of our actions: non predictive information is useless to us. Surprisingly most of the information we collect over a long period of time is non predictive, so that isolating the predictive information must extract from the sensory stream those features that are relevant for behaviour. Definition of learning: finding a generalised model that explains or describes a set of observations. Why generalised? Because we do not want to have an overfitted approximation of the data: Vapnik (1998) states that an animal can gain selective advantage not from its performance on the training data but only from its performance at generalisation. Learning a model is also equivalent to encoding the data produced by it (Rissanen, 1989), thus predicting and compressing are dual problems. Complexity is an intuitive property ascribed to physical systems like turbulent flows, ferromagnets materials etc... Complexity can be defined in two manners: intrinsic complexity and algorithimic complexity. The latter complexity was defined by and states that a true random string cannot be compressed and hence requires a long description (Kolmogorov, 1965), yet the physical process that generates this string may have a very simple description. Intuitively and from now on we refer to the complexity of the underlying process and not to the description length of the string generated from the process. Bialek et al. (2001) proved that predictive information $I_{pred}(T)$ provides a measure of complexity of the model underlying a time series. For small observation times:

$$I_{pred}(T, T') = H(T) + H(T') - H(T + T') \tag{4.1}$$

but in the limit of large observation times:

$$I_{pred}(T) = \lim_{T' \to \infty} I_{pred}(T, T') = H_1(T) \tag{4.2}$$

$H(T)$ is the entropy computed on the signal $x(t)$ for $-T < t < 0$ denoted in short hand by x_{past}, $H(T')$ is the entropy of the signal $x(t)$ that will be observed in the future $0 < t < T'$ denoted in short hand by x_{future}. If the future and the past are statistically independent $P(x_{future}|x_{past}) = P(x_{future})$ and viceversa $P(x_{past}|x_{future}) = P(x_{past})$, then we cannot make any prediction: the random guess is the best choice to predict the future. All predictions are probabilistic and so if the past tells us something about the future (and viceversa) we can use the conditional distribution of the future events on the past data: $P(x_{future}|x_{past})$. Where the density $P(x_{future}|x_{past})$ has smaller entropy compared to the prior distribution $P(x_{past})$ means that there was a reduction in entropy and that the particular future event is more likely to happen. The average of this predictive information is defined as:

$$I_{pred}(T,T') = \sum_{past,future} P(future, past) \cdot \frac{P_{future,past}}{P(future)P(past)} \tag{4.3}$$

$$P(future, past) = P(future|past) \cdot P(past) \tag{4.4}$$

and can be rewritten as:

$$I_{pred}(T,T') =< log_2[\frac{P(future|past)}{P(future)}] > \tag{4.5}$$

$$= - < log_2 P(future) > - < log_2 P(past) > - < [-log_2 P(future, past)] > \tag{4.6}$$

where < ... > denotes the average over the joint distribution of the past and the future. Because the elements of the equation are all entropies we can rename the variables as:

- $- < log_2 P(future) >= H(T')$
- $- < log_2 P(past) >= H(T)$
- $- < log_2 P(future, past) >= H(T,T')$

thus using the new variables in 4.6 gives us the equation in 4.1. What the mutual information tell us? $I_{pred}(T,T')$ is either the information that a data segment of duration T provides about the future length T' or the information that a data segment of duration T' provides about the immediate past of duration T. Now the entropy of a time series is proportional to its duration asymptotically, so that $\lim_{T\to\infty} H(T)/T = H_0$ thus entropy is an extensive quantity and predictability only depends on H_1 because:

$$I_{pred}(T,T') = H_0 \cdot T + H_1(T) + H_0 \cdot T' + H_1(T') - H_0 \cdot (T + T') - H_1(T + T') \tag{4.7}$$

and so $I_{pred}(T,T') = H_1(T) + H_1(T')$. Giving that:

$$H(T) = H_0 T + H_1(T) \tag{4.8}$$

$$\lim_{T\to\infty} H(T)/T = H_0 \tag{4.9}$$

$$H_1(T) \geqslant 0 \tag{4.10}$$

$$\lim_{T\to\infty} H_1(T)/T = 0 \tag{4.11}$$

and extending the future forwards toward infinity $T' \to \infty$ or the past towards minus infinity $T' \leftarrow -\infty$ the predictive information becomes:

$$I_{pred}(T) \quad = H_1(T), T' \to \infty \tag{4.12}$$

$$I_{pred}(T) \quad = H_1(T), T \leftarrow -\infty \tag{4.13}$$

This equality states that there is symmetry between prediction and postdiction $I_{pred}(T,T') = I_{pred}(T',T)$ but also that the predictive information at time $t = T$ gives us the same amount of information about the history of our observation as well as the same amount for the future ones that will start from the present time.

4.3.4 How much information is required for prediction?

As we observe a time series for a long time T, we accumulate data which is measured by the entropy $H(T)$, and for T that goes to infinity $H(T) \simeq H_0 T$. Because the predictive information cannot grow linearly with time, only a small fraction of it is relevant for prediction:

$$\lim_{T \to \infty} \frac{PredictiveInformation}{TotalInformation} = \frac{I_{pred}(T)}{H(T)} \to 0 \tag{4.14}$$

although we collect data in proportion to our time T, a smaller and smaller fraction of this information is useful in the problem of prediction. Nevertheless this property is true if the model that generates the time series has not changed its parameters, but what happens if the organism or somebody else "a deus ex" changes one of the parameters? The organism will experience a discontinuity in the predictive information that indicates a novelty or better a surprise.

4.3.5 Predictive information and model complexity

In the regime of infinite observation time $T \to \infty$, $I_{pred}(T)$ can:

- remain finite as $H_1 = h_1$.
- grow logarithmically $H_1 = h_1 + k \cdot log(T)$.
- grow as a fraction power law $H_1 = h_1 + h_2 \cdot T^{\alpha}$.

The first possibility $\lim_{T \to \infty} = h_1$, means that no matter how long we observe, we gain only a finite amount of information.

The second possibility $\lim_{T \to \infty} = k \cdot log(T)$, means that future observations depend on far distant past ones: the model that generates the time series has a number of finite parameters. The coefficient of the divergence k counts the number of parameters of the model.

The third possibility $\lim_{T \to \infty} \propto T^{\alpha}$, means that the underlying model has infinite parameters. Estimation of the sub-linear component can be achieved using non-linear regression methods or using evolutionary fitting.

4.3.6 Prediction and compression are related

Suppose now that one of our agents is deprived of his output with the environment so that it can only observe a set of data: $x_1, x_2, ..., x_N$. When we can say that the agent has learned? When the agent is able to predict the next observation x_{N+1} in case of 1 step prediction. The more an agent knows the more accurate the prediction is about x_{N+1} and the fewer bits are required to describe the difference or error from the previous observations. The average length of code required to describe the point x_{N+1} given the previous history:

$$l(N) = - < log_2(P(x_{N+1}|x_1, x_2, ..., x_N)) >_{P(x_1,...,x_N,x_{N+1})} \, bits \tag{4.15}$$

is the averaged conditional probability over the joint distribution of all the N+1 points. Remembering that the average code that describes a random sequence of N samples is the entropy $H(N)$ of that sequence, we can write:

$$l(N) = H(N+1) - H(N) \approx \frac{\partial H(N)}{\partial N} \tag{4.16}$$

We learn more when we use a smaller description for the time series. We can define a learning curve that measures the cost of encoding the next sample. The ideal encoding's length can be known if the agent observes the stream of data for a infinite time: $l_{ideal} = \lim_{T\to\infty} l(N)$, thus the learning curve is the difference between the actual code length and the ideal length:

$$\Lambda \equiv l(N) - l_{ideal} = \frac{\partial I_{pred}(N)}{\partial N} \tag{4.17}$$

The learning curve is the derivative of the predictive information and quantifies the information learned so far, if zero means that the optimal description code of the time series is reached.

4.3.7 Entropy reduction measure in learning agents

Predictive learning can be reduced to a probabilistic model and reformulated in terms of Bayesian learning. A simple model can be formulated using random discrete variables Y, X and W. Y is a binary random variable that represents the distal signal ($Y = 0, 1$) and X is a binary random variable that represents the reflex signal ($X = 0, 1$) where $Y = 1$ means that the distal signal was active and $Y = 0$ was not active. In an open loop case when the agent cannot feedback his actions to the environment, I can suppose that the reflex has the same probability of being present and absent $P(X = 0) = 0.5$ and the same condition applies to the distal signal $P(Y = 0) = 0.5$.
Using a non symmetric distribution like $P(X = 0) = 0.58$ implies the presence of a bias for the reflex to appear and indicates that the agent will do much work to compensate for that.

When the organism is regulating in a closed loop, a perfect regulator achieves an entropy reduction of the reflex as mentioned previously:

$$P(X = 0) = 1 \rightarrow H(X) = 0 \tag{4.18}$$

An imperfect regulator will be identified on the contrary by:

$$0 < P(X = 0) < 1 \rightarrow H(X) > 0 \tag{4.19}$$

A similar measure of the effectiveness of regulation could be the expectation of the variable X:

$$E(X) = \sum_{i=0}^{1} x_i \cdot p(x_i) \tag{4.20}$$

Perfect regulation implies that $E(X) = 0$, whereas imperfect regulation implies $E(X) \neq 0$ because the expectation $E(X)$ can be positive but also negative.

To be more clear, I can consider a better discretisation of the input space: $X = \{-1, 0, 1\}$ mapping in this case a negative error, a zero error, and a positive error. If before learning X has a uniform distribution like:

$$p(X_{before}) = \{1/3, 1/3, 1/3\} \tag{4.21}$$

$$\begin{aligned} E(X_{before}) &= -1 \cdot \frac{1}{3} + 0 \cdot \frac{1}{3} + 1 \cdot \frac{1}{3} = 0 && (4.22) \\ H(X_{before}) &= -3 \cdot \frac{1}{3} log_2(1/3) = log_2(3) = 1.5850\ bits && (4.23) \end{aligned}$$

But after learning or an equivalent successful perfect regulation where $p(X) = \{0, 1, 0\}$

$$\begin{aligned} E(X_{after}) &= -0 + 1 \cdot 1 + 0 = 1 \qquad (4.24) \\ H(X_{after}) &= log_2(1) = 0\,bits \qquad (4.25) \end{aligned}$$

Why is this so? Because entropy is a concave function of the distribution function p, whereas estimation is not able to distinguish between the two different steps of estimation. In my predictive performance, I have used the entropy to estimate the predictive performance because of the properties of entropy like non-negativity, concavity and chain rules for mutual information. However one does not have to exclude the expectation as a potential candidate for other useful purposes.

Intuitively if X and Y are causally dependent the predictive controller can achieve optimal prediction, whereas if if X and Y are only statistically dependent it will achieve sub-optimal prediction.

When the agent experiences the world using his innate reflexes, it can observe that $P(X|Y)$ the probability of observing the reflex X is dependent on the probability of observing Y, in order words X and Y are not conditionally independent (if they were independent $P(X|Y) = P(X) * P(Y)$).

If the agent does not use the distal signal $w_1 = 0$, it will observe that $P(X = 1|Y = 1) = k1$ and that the $P(X = 0|Y = 0) = k2$ is possibly high. Learning is achieved when the agent selects the best action so that $P(X = 1|Y = 1) < k1$, an important fact of predictive learning is although it is desirable that $P(X = 0|Y = 0) > k2$ it is not possible to do that because of the impossibility of the correlator to evaluate the pairing of a missing distal event with a missing reflex event.

This problem of the correlation of missing events is a weakness in many learning algorithms, but there have been some new models, like the one developed by Ian Glascher who is testing a dual model based on the Wagner-Rescorla equation, which consider the positive rewarded outcome complementary to the negative rewarded outcome. In other words the model also considers what did not happen after the agent made a particular choice. Predictive learning based on correlation therefore wants to choose actions so that the distribution of the events is:

- before learning $P_{before}(X = 1|Y = 1) = 0.6$,$P(X = 0|Y = 1) = 0.4$

- after learning $P_{after}(X = 0|Y = 1) = 0.8$, $P(X = 1|Y = 1) = 0.2$

So the change in the distribution of $P_k(X = 0|Y = 1)$ is an index of the agent predictive power. However I need also to consider how well the agent has learnt to avoid the reflex or equivalently to regulate itself. I need to find an information measure which combines:

- the predictive power of the agent

- the regulatory power of the agent

Since I want to use entropy for its attractive properties of concavity and non-negativity, I can revisit the concept of conditioned entropy and mutual information and see if they suit our purposes. Conditioned entropy:

$$\begin{aligned} H(X|Y) &= -\sum_x \sum_y p(x,y)log(p(x|y)) && (4.26) \\ H(Y|X) &= -\sum_x \sum_y p(x,y)log(p(y|x)) && (4.27) \\ H(X|Y) &= H(X,Y) - H(Y) && (4.28) \\ H(Y|X) &= H(X,Y) - H(X) && (4.29) \\ & && (4.30) \end{aligned}$$

Where the joint entropy $H(X,Y)$ is formulated as:

$$H(X,Y) = -\sum_x \sum_y p(x,y)log(p(x,y)) \qquad (4.31)$$

and the mutual information as:

$$\begin{aligned} I(X,X) &= 0 && (4.32) \\ I(X,Y) &= I(X,Y) && (4.33) \\ I(X,Y) &= H(X) - H(X|Y) && (4.34) \\ I(X,Y) &= H(X) + H(Y) - H(X,Y) && (4.35) \\ I(X,Y) & \quad \geqslant 0 && (4.36) \end{aligned}$$

In the next section I am going to evaluate which measure captures the learning performance of the agent considering typical probability distributions before and after learning. The joint density can be represented as a matrix of 2 by 2 elements because in this case the variables X,Y are binary. The table is constructed from the experimental data and must fulfil the properties of probability distributions:

- the integral of the joint distribution: $\sum_X \sum_Y p(x,y) = 1$
- the integral of marginal distribution X: $\sum_X p(x) = 1$
- the integral of marginal distribution Y: $\sum_X p(y) = 1$

The Table 4.1 shows a typical density distribution before learning, considering the assumption of a uniform distribution for the reflex X and distal events Y, and that the agent has not yet learned to avoid the undesired state $x = 1$ using the distal event $y = 1$, thus $p(y = 1, x = 1) = 0.4$.

From the table 4.1, I can compute the entropies in bits:

- $H(X) = 1\ bit$
- $H(Y) = 1\ bit$
- $H(X,Y) = 1.7219\ bits$
- $H(X|Y) = 1.7219 - 1 = 0.7219\ bits$

Table 4.1: Entropy values before learning.

XY	$p(y=0)$	$p(y=1)$	$p(X)$
$p(x=0)$	0.4	0.1	0.5
$p(x=1)$	0.1	0.4	0.5
$p(Y)$	0.5	0.5	1.0

- $I(X,Y) = 1 + 1 - 1.7219 = 0.27810\ bits$

The Table 4.2 shows the density distribution after learning was achieved: the agent swaps the rows in the column of $y = 1$:

Table 4.2: Entropy values before learning.

XY	$p(y=0)$	$p(y=1)$	$p(X)$
$p(x=0)$	0.4	0.4	0.8
$p(x=1)$	0.1	0.1	0.2
$p(Y)$	0.5	0.5	1.0

From the table 4.2, I can compute the entropy measures:

- $H(X) = 0.72193$
- $H(Y) = 1$
- $H(X,Y) = 1.7219$
- $H(X|Y) = 1.7219 - 1 = 0.7219$
- $I(X,Y) = 1 + 0.72193 - 1.7219 = 0.00003$

The mutual information after learning has been reduced to a very small number because $H(X,Y)$ is invariant to row or column permutations of the conditioned probability, but it is not the case for the marginal distributions $H(X)$ and $H(Y)$ that are changed because in this case $p(X = 0) = 0.8$ and $p(X = 1) = 0.2$. This means that the agent has learned to avoid the undesired state $X = 1$. An agent with perfect learning has a probability distribution as in Table 4.3:

When perfect learning is achieved the agent is always keeping the desired state $p(X = 0) = 1.0 \rightarrow H(X) = 0$, no matter what the distal event was $p(x = 0, y = 0) = p(x = 0, y = 1) = 0.5$, hence $H(X,Y) = 1$ and $H(Y) = 0.5$. However the entropy measure does not distinguish how to equivocate between an agent that learned "a good thing" from the one who learned "a bad thing". For example in Table4.4, where the agent has swapped $p(x = 0|y = 0)$ with $p(x = 1|y = 0)$, it produces exactly the same values for the mutual information and the other measures but means that the agent has learnt to produce an action that, when the distal is present, evokes a reflex.

Table 4.3: Entropy values before learning.

XY	$p(y=0)$	$p(y=1)$	$p(X)$
$p(x=0)$	0.5	0.5	1.0
$p(x=1)$	0.0	0.0	0.0
$p(Y)$	0.5	0.5	1.0

Table 4.4: Equivocation table.

XY	$p(y=0)$	$p(y=1)$	$p(X)$
$p(x=0)$	0.1	0.1	0.2
$p(x=1)$	0.4	0.4	0.8
$p(Y)$	0.5	0.5	1.0

Therefore when considering learning, I need to have a look at the action policy: how the agent chooses an action to achieve the desired state. In our simulation the agent is properly wired so that it will compensate for the distal event but an improperly wired agent or an agent with a wrong action policy may reach non desired states while learning. Therefore for a more general approach, I need to consider either the action policy or restrict the probability. Intuitively, the combined system contains $H(X,Y)$ bits of information: we need H(X,Y) bits of information to reconstruct its exact state. If we learn the value of Y, we have gained $H(Y)$ bits of information, and the system has $H(X|Y)$ bits of uncertainty remaining. $H(X|Y)=0$ if and only if the value of X is completely determined by the value of Y. The Bayes theorem can be used to calculate the conditioned probabilities $p(y|x)$ from $p(x|y)$. This is more easy than computing $p(x|y)$ in our simulation because of the causal temporal relation between y and x (y follows x).

$$P(Y=0|X) = \frac{P(X|Y=0)\cdot P(Y=0)}{P(X|Y=0)+P(X|Y=1)} \tag{4.37}$$

$$P(Y=1|X) = \frac{P(X|Y=1)\cdot P(Y=1)}{P(X|Y=0)+P(X|Y=1)} \tag{4.38}$$

$$\tag{4.39}$$

However this is not necessary if the estimation is made offline.

4.4 Industrial applications

There are two industrial applications of social systems in the market right now. One is the Kiva System [1], an automatic warehouse solution and the other is the Eporo system developed by Nissan. The Kiva System is a group of robots which are placed in a warehouse and can optimize the order fulfilment. The Kiva robots are able to communicate with each other and

[1] http://www.kivasystems.com/

with a sort of control tower which assigns priorities to each robot. The system is a very clear implementation of the advantages of using a social approach to a traditional warehouse task. The Eporo [2] system is essentially a swarm behaviour implemented in concept cars. The main goal is to use the school fish behaviour to avoid accidents in high density traffic. In this application there is no central controller and thus it is more distributed but the communication is more on the action level. The author is confident that in the future there will be more and more social robotics implementation in the industry.

[2] http://www.nissan-global.com/EN/NEWS/2009/_STORY/091001-01-e.html

Chapter 5

Appendix

5.1 ICO learning parameters

5.1.1 ICO learning

The input correlation learning rule by Porr and Wörgötter (2006) is a Heterosynaptic learning rule, it is unsupervised and performs a correlation between a predefined reflex signal (x_0) and a reflex predicting signal (x_1). Hence, this learning algorithm identifies and exploits causalities between temporal sequential signals.

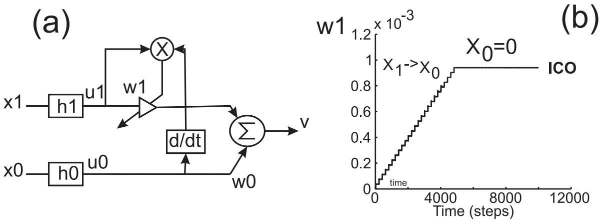


Figure 5.1: Figure (a) shows the ICO learning basic block composed by 2 inputs x_0, x_1 filtered by h_0, h_1 and the output v. Figure (b) shows the weight change of w_1 during time. At the beginning $w_1 = 0$, then for 5000 simulation steps x_1 anticipates x_0 but after that w_1 stabilize to $1 \cdot 10^{-3}$.

Figure 5.1 shows the ico learning block which has two inputs x_0,x_1 from the agent's sensor that are filtered by bandpass filters h_0, h_1. The transfer function h is a bandpass which transforms a d-pulse input into a damped oscillation and is specified in the Laplace-domain:

$$h(t) \leftrightarrow H(s) = \frac{1}{(s+p)(s+p*)} \tag{5.1}$$

where $p*$ represents the complex conjugate of the pole $p = a + ib$. It is important to note that such a bandpass is only stable if its pole-pair is located on the left complex half-plane, otherwise an amplified oscillation is obtained.

$$
\begin{aligned}
h(t) &= \frac{1}{b}e^{at}sin(bt) && (5.2)\\
a &= -\pi\frac{F}{Q} && (5.3)\\
b &= \sqrt{(2\pi F)^2 - a^2} && (5.4)
\end{aligned}
$$

F is the oscillation frequency and Q the quality factor. The damping characteristic of the filter is reflected by Q (see Appendix A). Filtered signals $u_i(t)$ are transferred with weight w_i to the output neurons. In the output neuron the output $v(t)$ is calculated by summing up all incoming signals according to their weights:

$$v(t) = \sum_{k=0}^{1} w_k u_k \qquad (5.5)$$

which represent the input for the motor system. The unsupervised character of the ICO learning rule is reached by the synaptic weight w_1 to be adapted by the weight change rule:

$$\frac{d}{dt}w_1 = \mu u_1 \frac{du_0}{dt} \qquad (5.6)$$

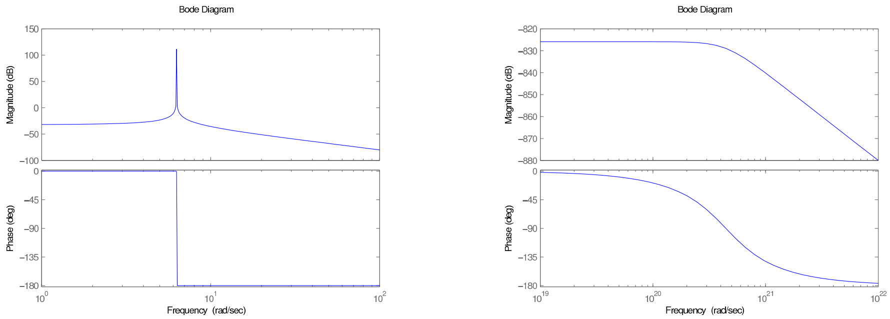
The weight change is dependent on the derivative of the reflex input signal u_0, the input signal u_1 and a learning rate μ. The learning rule has been shown to be useful for avoidance and attraction mechanisms and has fast and stable convergence (Porr and Wörgötter, 2006).

The filter response h is parametric in Q. If $Q > \infty$ it means that:

- $a = 0$
- $b = 2\pi f$

therefore the filter response becomes $H(s) = \frac{1}{(s^2-(2*\pi*f)^2)}$ that is a resonator with frequency $2\pi f$, see Fig. 5.2(a). When Q approaches 0, $Q \to 0$ this is the result: $H(s) = \frac{1}{(s^2+2as+2a^2)}$ a trinomial term with infinite cutoff frequency, see Fig. 5.2(b).

When $Q = 1/\sqrt(2)$ the filter has a maximally flat response without any overshoot. When $Q \leq 0.5$ the filter has no undershoot in the impulse response.



(a) Bode diagram of h filter when $Q > \infty$

(b) Bode diagram of h filter when $Q \to 0$

Figure 5.2: Bode diagrams for different Q values.

5.2 Simulation details

World size is directly proportional to the agent and food area. The agent size is $[W_r, H_r] = 2.50U \cdot 2U$ with an area $A_{agent} = 5U^2$, where U is the unit of measure used in the simulations. Food sources are regular polygons composed by $e_{food} = 32$ edges with an apothem of $ap_{food} = 3U$ corresponding to an area of

$$A_{food} = ap_{food}^2 \cdot e_{food} \cdot tan(\frac{\pi}{e_{food}}) = 28.36U^2. \qquad (5.7)$$

It means that only $N_{max,food} = 9$ robots can lie on the food disk boundary, because

$$A_{max,agents} = \frac{S^2 \cdot e_{food}}{4 \cdot tan(\frac{\pi}{N_{max,food}})} = 24.73U^2 \qquad (5.8)$$

where $S = 2.0U$. The area of the world is proportional to the areas of agents and food disks, and on their numbers.

$$A_{world} = (K_1 \cdot A_{food} \cdot M) + (K_2 \cdot A_{agent} \cdot N) + (K_2 \cdot A_{agent} \cdot N) \qquad (5.9)$$

where $K_o = 50, K_a = 60$. The world is a square with a side of $L = \sqrt[2]{A_{world}}$. The robot boundary surface is delimeted by the following points: $< -1.25, -1 >, < 1.25, -1 >, < 1.25, 1 >, < -1.25, 1 >$

Robot parameters:

- differential 2 wheeled driving system
- 2 infrared sensors: placed in front of the robot, one left one right, oriented with $\alpha(l, r) = \pm 45°$, an aperture of $IR_{field} = 30°$, a range of $range_{IR,reflex} = 2.5U$ and $ray = 3$.
- 2 infrared sensors: placed in the back of the robot, one left one right, oriented with $\alpha(l, r) = \pm 160°$, an aperture of $IR_{field} = 60°$, a range of $range_{IR,reflex} = 2.5U$ and $ray = 3$.

- 2 circular RGB cameras $ccam_{i,proximal}$ with a visibility range of $10.0U$: placed in front of the robot, one left one right, $h = 2.0U$ from the ground, oriented with $\alpha = \pm 30°$, an aperture of $field_{proximal} = 60°$ and a resolution of $ccam_{res} = 16$ pixels.

- 2 circular RGB camers $ccam_{i,reflex}$ with a visibility range of $8.0U$: placed in front of the robot, one left one right, oriented with with $\alpha = \pm 30°$, aperture of $field_{proximal} = 60°$ and a resolution of $ccam_{res} = 16$ pixels.

Sensors' and actuators' response functions are modeled on the Alice robot by EPFL[1] into the ENKI simulator environment. Motors have a normalised speed of $[v_{min}, v_{max}] = [-1, 1]$ within an inversely proportional motor noise to the speeds in $[-1, 1]$, the noise is of the order of $\pm 5\%$ for the maximum speed, for $\pm \frac{3U}{t}$ it is $\pm 10\%$ The objects in the world have different colours that represent their type. The agent can see colors using their RGB circular cameras. A color is represent by 3 components: red,blue and green as a triplet $< r, g, b >$ where $r, g, b \subset [0.0, 1.0]$. For example red is represented by $< 1.0, 0.0, 0.0 >$. The rule for the colors are:

- the agent's color is:$color(agent) = < 1.0, 0.0, H_{sated}(t) >$ as described in eq.3.16. Thus an agent has a fixed red component and a blue component that expresses its internal state of satedness.

- the food place's colour for the unlimited case is:$color(food_{infinite}) = < 0.0, 1.0, 0.0 >$

- the food place's colour for the limited case is:$color(food_{finite}) =< 0.0, 1.0, q(t) >$ as in eq.3.30.

- an object with green component is a food source.

According to the ICO terminology the agent's configuration is:

- left and right infrared sensors are the reflex for obstacles: world's walls, other agents, food sources

- left and right long range camera filtering the red component are the proximal for obstacles: other red agents, or food sources before learning

- left and right long range camera filtering the blue component are the distal signals for agent attraction: other agents consuming food.

- left and right short range camera filtering the blue component are the proximal signals for agent attraction: other agents consuming food.

- left and right long range camera filtering the green component are the distal signal for food attraction.

- left and right short range camera filtering the green component are the proximal signal for food attraction.

[1] Ecole Polytechinique Fdrale de Lausanne

The back left and right infrared sensors are useful when the robot is going backwards and bumps into an obstacle: in that case the speed is inverted. The left,right camera inputs must be transformed from a vectorial format to a scalar one for the input synapses. Every circular camera produces a matrix $Vcam_{NPIXELS,3}$ where $npixels = 16$, the scalar filtered input is computed as:

$$vcam = \sum_{i=1}^{npixels} Vcam_{i,fcol} \tag{5.10}$$

where $fcol$ is the chosen color component to be filtered.
The ICO network topology is described step by step, adding the relative features: obstacles avoidance first, food attraction and then agent with food attraction. Tthe configuration used for the network parameters follows.

5.2.1 Obstacle avoidance

The obstacles have a red component therefore in eq. 5.10 $fcol = 0$. The agent must avoid obstacles: it means that the left input is connected with the right output with a negative weight, and the left input is connected with the left output with a negative weight. In this way when an obstacle is on the left side the robot will go back turning left.

- IR sensors: $W_{proximal,L} = -0.9$,$W_{proximal,R} = -1.0$
- short range camera: $W_{distal,L} = 4.0$,$W_{distal,R} = -4.2$
- Filter parameters: $F_{proximal} = 0.5$,$F_{distal} = 0.6$,
  $Q_{proximal} = \sqrt(2)$,$Q_{distal} = \sqrt(2)$ Q is chosen to have a maximum flat response

5.2.2 Food attraction parameters

The food sources have a green component therefore in eq.5.10 $fcol = 1$. The agent must approach the food, therefore the left and right synaptic input values must be equal: it means that the difference between the left and right input must be 0.

- $W_{proximal,L} = 2.6$,$W_{proximal,R} = -2.8$
- $W_{distal,L} = 3.0$,$W_{distal,R} = -3.2$
- Filter parameters: $F_{proximal} = 0.6$,$F_{distal} = 0.6$,
  $Q_{reflex} = \sqrt(2)$,$Q_{proximal} = \sqrt(2)$

5.2.3 Agent with food attraction parameters

The agents have a red and blue component therefore in eq.5.10 $fcol = 0, 2$. The agent must approach the other agent, therefore the left and right synaptic input values must be equal: it means that the difference between the left and right input must be 0.

- $W_{proximal,L} = 3.0$,$W_{proximal,R} = -3.2$
- $W_{distal,L} = 3.0$,$W_{distal,R} = -3.2$

- Filter parameters: $F_{proximal} = 0.6$,$F_{distal} = 0.6$, $Q_{reflex} = \sqrt(2)$,$Q_{proximal} = \sqrt(2)$

5.2.4 Hysteresys effect

If an agent is in proximity of an acute angle (it could be an angle of the scenario or a particular spatial agent configuration) without the hysteresys it will starve turning left and right, within the hysteresys the responses will be delayed so it will turn first left and after some time right. The weight connections are asymmetric to balance the outputs (using the same weights makes the robot run in circular pathways). Parameters are:

- $W_{self,L} = 0.4$,$W_{self,R} = 0.4$
- $W_{left,right} = -0.42$,$W_{right,left} = -0.3$

The bias for the network is: $bias = 2.4$.

5.2.5 Physical engine and kinematic model

Every object in the world has the following physical properties:

1. position is 2 dimensional vector
2. height: the height of the object, used for interaction with robot's sensors.
3. angle: the orientation of the object in the world, standard trigonometric
4. vector speed: The speed of the object
5. angle: the orientation of the object in the world, standard trigonometric orientation.
6. angular speed: the rotation speed of the object, standard trigonometric orientation.
7. mass: the mass of the object. If below zero, the object can't move (infinite mass).
8. static friction threshold: the static friction threshold of the object. If a force is smaller than it, the object will not move
9. viscous friction tau: the viscous friction time constant. Half-life of speed when object is free. If lower than timestep, speed is forced to zero.
10. viscousMomentFrictionTau: the viscous friction moment time constant. Half-life of angular speed when object is free. If lower than timestep, angular speed is forced to zero.
11. collisionAngularFrictionFactor: upon collision with static objects. The amount of rotation transmitted to the moving object. If zero, moving object slides over static one. If one, moving object is fully rotated.

The function that updates the state of the object is:

```
void PhysicalObject::step(double dt)
{
        pos += speed * dt;
        angle += angSpeed * dt;
        angle = normalizeAngle(angle);
        // TODO : optimise this using ExpDecay from external math lib !
        if (viscousFrictionTau < dt)
        {
                speed = 0.0;
        }
        else
        {
                double factor = (viscousFrictionTau - dt * 0.5) /
                                (viscousFrictionTau + dt * 0.5);
                speed *= factor;
        }
        if (viscousMomentFrictionTau < dt)
        {
                angSpeed = 0;
        }
        else
        {
                //std::cerr << "0   f:" << angSpeed << std::endl;
                double factor = (viscousMomentFrictionTau - dt * 0.5) /
                (viscousMomentFrictionTau + dt * 0.5);
                angSpeed *= factor;
                //std::cerr << "angSpeed:" << angSpeed << std::endl;
                //std::cerr << "factor:" << factor << std::endl;
        }
}
```

The kinematic model used for the differential driving system is fairly simple:

```
void Alice::step(double dt)
{
        double realLeftSpeed, realRightSpeed;

        // applied inversely proportional motor noise

        realLeftSpeed = 1 * leftSpeed * (0.95 + random.getRange(0.05));

        // same as above
        realRightSpeed = 1 * rightSpeed * (0.95 + random.getRange(0.05));

        // forward component
        double forwardSpeed = (realLeftSpeed+realRightSpeed) / 2;
        double wheelDist = 1.9;
        // Khepera code:
        speed = Vector(
        forwardSpeed * cos(angle + angSpeed * dt * 0.5),
        forwardSpeed * sin(angle + angSpeed * dt * 0.5));
```

```
// angle
angSpeed += (-realLeftSpeed+realRightSpeed) / wheelDist;
```

For collision detection and collision interaction, detailed information is provided in the source code.

5.2.6 Simplified social simulator

There are $N = 2, 4$ and $M = 2$ food sources. The agent's area $A_{agent} = 5U^2$, where U is the unit of measure used in the simulations. Food source's area $A_{food} = 28.36U^2$. The area of the world is proportional to the areas of agents and food disks, and on their numbers.

$$A_{world} = (K_1 \cdot A_{food} \cdot M) + (K_2 \cdot A_{agent} \cdot N) + (K_2 \cdot A_{agent} \cdot N) \tag{5.11}$$

where $K_o = 50, K_a = 60$. The world is a square with a side of $L = \sqrt[2]{A_{world}}$. For $N = 2$ and $M = 2$: $A_{world} = 3436U^2$ and $L = 58.617$.

5.2.7 Screenshots

In figures 5.3,5.4 there are some screenshots of the simulator environment: the graphic interface makes use of the OpenGL API to draw the scene.

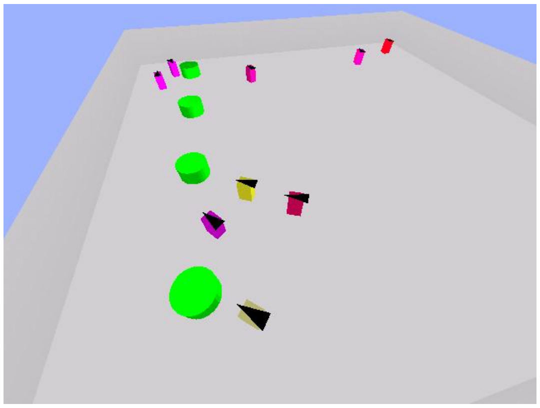


Figure 5.3: A simulation with 10 agents and 4 food disks.Yellow agents signals the food presence, violet agents are sated, red agents are starving. Food disks are the green disks.

5.2.8 Simulation details for information flow

The world is a toroidal square of 300x300 units (Um), the agent has a diameter of 10 Um, the reflex antennas have a range of 40 Um, the predictor antennas have a range of 60 Um, every food disk has a diameter of 20 Um, the agent consumes food after 30 time steps. Every food disk starts with 100 food units and, if depleted, is reset after 5 time steps. To compute the entropy, the input space is discretised into 4 equally spaced bins and normalised in the range [-1,1] both for the predictor U_1 and the reflex U_0 signal, the output signal Z is discretised in 8 directions.

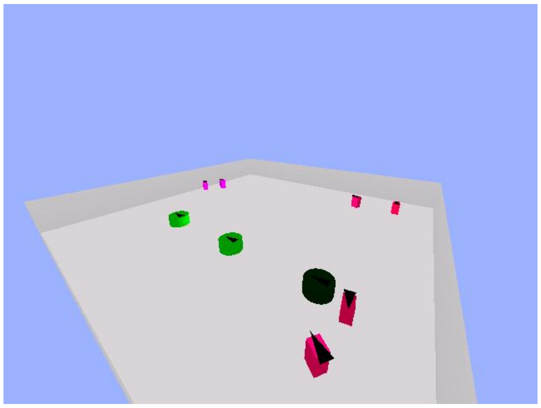


Figure 5.4: A simulation with 10 agents and 4 food disks.Yellow agents signals the food presence, violet agents are sated, red agents are starving. Food disks are the green disks, when the food disk is exhausted tit urns black and agents are not attracted.

5.2.9 Simulation details for Q-learning robot

The world is a square of 200x200 units (Um). The world can be wrapped in toroidal coordinates or not. If the world is toroidal that means there are no walls and thus no collisions, whereas when the world is non toroidal the robot bounces off the wall but does not take any negative reward. In each learning session there are 20 obstacles placed randomly in the world. The radius of the agent is 6 Um. The radius of the obstacle is 20 Um. The robot turns angle in multiples of $\theta = 0.4 rads$.

5.3 Weight clustering

A better way to quantify the seekers and parasites in the social system is to do a clustering analysis based on the two dimensional space of the weight. Each agent can be represented as a point in a two dimensional space where the x-axis represents the weight of the food seeking behaviour and the y-axis represents the weight of the parasitic seeking behaviour. Initially because all the agents have the same predictive weight, there will be N points in the same location, during the specialization process there will be a cloud of points. The cloud will have a different shape according to the proportion of seekers and parasites. For instance seekers will be described as points spread on an horizontal cluster, whereas parasites will be spread on a vertical cluster. The larger the variance, the better the specialization, therefore PCA can be used to quantify the degree of specialization by looking at the main component of the space. For example in Fig.5.5 I have plotted the evolution of the points during learning for two different cases:

- $N = 20$ agents and $M = 4$ food sources: at the end of the simulation there are 16 parasites identified by the horizontal cluster
- $N = 20$ agents and $M = 18$ food sources: at the end of the simulation there are 16 seekers identified by the vertical cluster

When the system differentiates in half seekers and half parasites, there is only one cluster with a circular shape.

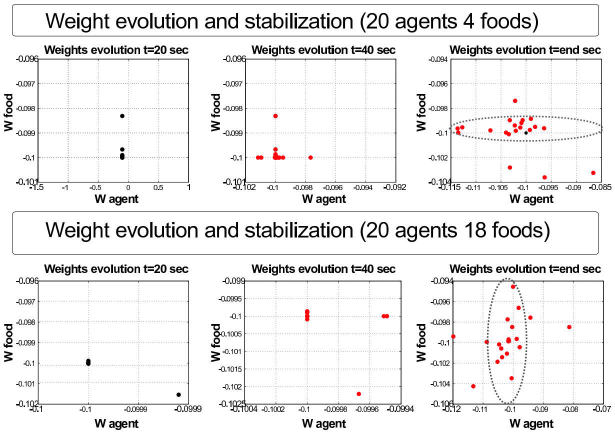


Figure 5.5: Weight evolution in a two-dimension space for two important cases: on the first row there are 20 agents for only 4 food sources, on the second row there are 20 agents for 18 food sources. Each plot contains on the x-axis the synaptic weight for the seeking behaviour and on the y-axis the synaptic weight for the food seeking behaviour.

5.4 Information theory

5.4.1 Variety

Variety was defined by Ashby as the number of different elements in a set. A set composed by its alphabet has maximum variety of: $A = \alpha_1, ..., \alpha_n$ then the variety is n: $V_A = n$ Variety can also be expressed as the number of bits required to describe the set, thus $VL_A = \log_2 V_A$. Example: $V_A = 2^4$ then $VL_A = 4$, meaning that an alphabet of 4 different symbols can be encoded as a string of 4 binary digits. So the logarithm in base 2 of variety is equivalent to the entropy measure which is the average length of the shortest description of the corresponding random variable (Shannon and Weaver, 1949): the expected length $L(C)$ of a source code $C(x)$ for a random variable X with probability mass function $p(x)$ is given by:

$$L(C) = \sum_{x \in X} p(x) l(x) \tag{5.12}$$

where l is the length of the codeword associated with x. Then an optimal code L^* is bounded by:

$$H_A(X) \leq L^* < H_A(X) + 1 \tag{5.13}$$

where A is the afore mentioned alphabet. A binary alphabet is composed by 2 symbols: $A = 0, 1$.
If the set of disturbances D is $D = \{d_1, d_2, d_3, d_4\}$, then $V_D = 4, VL_D = 2$.
If the set of disturbances D is $D = \{d_1, d_2, d_3, d_3\}$, then $V_D = 3, VL_D = 1.58$.

5.4.2 Entropy

Entropy estimates the average uncertainty (or information) of a variable (Shannon and Weaver, 1949) The most common form to compute the discrete entropy of a stochastic variable X is calculated according to:

$$H(X) = -\sum_{i=1}^{N} p(x_i) \cdot log_2 p(x_i) \geq 0 \tag{5.14}$$

where the probability $p(x_i)$ of X being in state x_i (there are in total N states) is estimated from the time series with one of the methods described in the following section. When the logarithm of base 2 is used, as in this case, the units are bits. The entropy is maxima, for a uniform distribution which means $p(x_i) = 1/N$ for $\forall i$, then $H(X) = log_2 N$

5.4.3 Mutual information

Mutual information measures the deviation from statistical dependence of two variables and is sensitive to **any** relationship between 2 variables, not only linear dependencies. The mutual information of two discrete variables, X and Y, can be expressed as:

$$MI(X,Y) = -\sum_{i=0}^{N_X} \sum_{j=0}^{N_Y} p(x_i, y_i) log_2 \frac{p(x_i p(x_j))}{p(x_i, y_j)} \tag{5.15}$$

The mutual information can also be expressed as:

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) \tag{5.16}$$

thus the mutual information is high if both X and Y have high entropy (high variance), and hare highly correlated (high covariance); it is zero if X and Y are statistically independent, and thus $p(x_i, y_j) = p(x_i)p(y_j)$. Similar to the difference between observed and true entropy, as a result of the finite size of the data sets, a difference exists between observed mutual information and true mutual information (Roulston,1999). The correction formula is:

$$MI_{true} \approx MI_{observed} - \frac{P_X P_Y - P_X - P_Y + 1}{2N} \pm \sigma_I \tag{5.17}$$

where P_X and P_Y are the number of states in which X and Y are discretised, N is the size of the time series (supposed equally long), and σ_I is the "logarithm in base two" corrected standard deviation of the estimate as explained in (Roulston, 1999) Eq.42. A good heuristic is to choose $N > 3P_X P_Y$. Mutual information is an index of statistical dependence and does not indicate causal relationships or directional information transfer. A candidate measure for the temporal dependence is then:

$$MI(\tau) = MI(X_t, Y_{t-\tau}) = H(X_t) + H(Y_{t-\tau}) - H(X_t, Y_{t-\tau})) \geqslant 0 \tag{5.18}$$

where τ denotes the delay. It is common practice to compute the mutual information for different taus τ_i and then find the $\max_{\tau_i} MI(\tau_i)$. Whereas the mutual information is symmetrical $MI(X,Y) = MI(Y,X)$ the temporal information is not $MI(X_t, Y_{t-\tau}) \neq MI(Y_t, X_{t-\tau})$.

5.4.4 Motor output entropy

The entropy $H(Z)$ is the uncertainty of the motor output or the average description necessary to encode the motor output. In our case the controller produces a continuous output angle in the range $z = [0, 360]$ degrees (see Section 3.5).

To compute the $H(Z)$ we have to discretise the angle z by choosing a bin width of $\Delta Z = 1$ degree for example. So the alphabet of $Z = \{1, 2, ..., 360\}$ contains 360 symbols and in term of entropies:

- If $H(Z) = 2$ bits, it means that the robot is able to steer with 4 different angles with 1 degree resolution.
- If $H(Z) = 8.5$ bits, it means the robot is using the full range of available steering angles. The robot could be generating a variety of different patterns such as circular motions, snake trajectories or a mix of the 2.
- If $H(Z) = 0$ bits, it means that the robot is heading always in the same direction.

The output entropy does not tell us anything about the effort of the controller, even a simple controller in a complex environment will produce a high $H(Z)$.

5.4.5 Retinal predictive entropy

The predictor is composed of a left and right retina arranged as a grid of N_{rf}xN_{rf} inputs, but to simplify the computation of the entropy we are going to use the retinal difference $x_{1,i,j}$ which is the difference between the left and right retinal input at position (i,j). Therefore $H(X_{1,i,j})$ is the uncertainty of the predictor input at position (i,j), whereas $H(X_{1,i,j})$ is the average entropy of the retinal differential input

$$H(X_1) = \frac{1}{N_{rf}^2} \cdot \sum_{i,j=1}^{N_{rf}} H(X_{1,i,j}) \tag{5.19}$$

5.4.6 The law of requisite variety for predictive learning

To apply the law of requite variety for a predictive controller we need to add another source of information D' before the disturbance D. The regulator will have an additional source of information

D' rather than having only the disturbance D. Thus the additional information that D' provides about the disturbance D is the mutual information between D and D', $I(D, D')$.

$$H(E) \geq H(D) - I(D, D') + H(R|D) - H(R) \tag{5.20}$$

If D' is not predictive of D, this means that D' and D are independent thus $I(D, D') = 0$ because the joint distribution $p(D, D') = p(D) * p(D')$. If Y is predictive of D, then $I(D, D') > 0$ thus allowing a reduction in the required entropy for the desired states of the regulator. The equation is consistent because :

$$H(D|D') = H(D) - I(D, D') \tag{5.21}$$

$$H(E) \geq H(D|D') + H(R|D) - H(R) \tag{5.22}$$

Thus, for instance if the organism has 4 possible actions $H(R) = 2$ and a variety of 8 disturbances $H(D) = 3$, if there is no predictive information the minimum variety we can reach for the state variable is $H(E) = 3 - 2 = 1$, and E must have at least a variety of 2. However if a predictive signal is available so that it provides more information about the disturbance $I(D|D') = 1$, the minimum variety that we can achieve with the state variable is $H_{min}(E) = 3 - 1 - 2 = 0$, that means we can achieve a perfect regulation to 1 state! The term $I(D, D')$ can also be considered as the predictive capacity of the agent or how much the agent has compressed the environment model. Supposing that agents have a limited memory then the mutual information cannot be used totally because compression is not perfect. The parameter $0 < \alpha < 1$ indicates the compressive capacity of the agent. Thus the minimum entropy for the desired states is increased:

$$H(E) \geq H(D) - \alpha I(D, D') + H(R|D) - H(R) > H(D) - I(D, D') + H(R|D) - H(R) \tag{5.23}$$

For example in our previous example if $\alpha = 0.5$ then the minimum entropy becomes $H_{min}(E) = 3 - 0.5 - 2 = 0.5$ bits.

5.5 Input correlation method

5.5.1 Cross correlation corrections

Cross correlation measure cannot be applied directly to variable time windows, thus correction terms based on the number of samples N must be included. Corrected cross correlation measures are: biased, unbiased, normalised. Biased estimate of the cross correlation function is:

$$R_{xy,biased}(m) = \frac{1}{N} R_{xy}(m) \tag{5.24}$$

Unbiased estimate of the cross-correlation function is:

$$R_{xy,unbiased}(m) = \frac{1}{N - |m|} R_{xy}(m) \tag{5.25}$$

Normalised cross correlation is:

$$c_{xy}(m) = \frac{1}{N-m} \sum_{1}^{N-m} \frac{x_i - \bar{x}}{\sigma_x} \frac{y_i - \bar{y}}{\sigma_y} \tag{5.26}$$

where $\bar(x)$ and σ_x denotes mean and variance of x, and m is a time lag.

5.5.2 Coherence function

In Laplace domain cross correlation is defined as:

$$C_{xy}(\omega) = (Fx)(\omega)(Fy) * (\omega) \tag{5.27}$$

where (Fx) is the Fourier transform of x, ω are the discrete frequencies $(-N/2 < \omega < N/2)$ and $*$ means complex conjugation. The cross spectrum $C_{xy}(w)$ is a complex number whose normalised amplitude:

$$L_{xy}(\omega) = \frac{| < C_{xy(w)} > |}{\sqrt{< C_{xx}(w) >} \sqrt{< C_{yy}(w) >}} \tag{5.28}$$

is called the coherence function and gives a measure of the linear synchronization between x and y as a function of the frequency ω. This measure is very useful when synchronisation is limited to some particular frequency band, as it's usually the case of EEG signals (for a review see Niedermeyer and da Silva (2004)).

5.5.3 Energy and power of digital signal

For the digital signal $c_d(m,k)$, where d is the synapse direction ($d = \{left, right\}$), m is the value of the cross correlation in the time window of index $k = 0, 1, ..$ with N_s samples:

- Energy: $E(c_d(k)) = \sum_{m=1}^{N_s} (c_d(m,k))^2$
- Power: $P(c_d(k)) = \frac{\sum_{m=1}^{N_s} (c_d(m,k))^2}{N_s}$

5.5.4 Alternative measures for analog signals

One can apply for instance synchronisation measures for EEG signals (for a review see Quiroga et al. (2002)) even if we are not interested to detect the (driver-response) relationships between signals. Indeed we already know that x_1 precedes x_0 in the avoidance case because this relation is a physical causal property of the environment. However these measures should equally work, with an extra computational cost:

1. linear measures:
   (a) normalized cross-correlation
   (b) coherence function
2. non-linear measures:
   (a) S,H,N
   (b) Mutual information and transfer entropy
   (c) Hilbert phase analysis
   (d) Wavelet phase analysis

According to these EEG studies it has been shown that cross correlation is less sensible than non-linear measures because EEG signals are produced by non-linear systems, so actual simulations are investigating if this property is true so too in our multi agent system.

5.5.5 Alternative measures for discrete time series

If we are dealing with discrete signals like spike trains, temporal information can be computed by: ISI distance, Rosendal distance,Euclidean distance, cross-correlograms, joint peristimulus PSH.

5.6 EMPASS: a parallel ABM simulator

This Chapter contains an introduction to parallel computing and explains why it is important to adopt a parallel approach to ABM with the recent boom in the market of parallel CPU and GPU. The approach used in this Thesis was to implement a parallel ABM engine to speed up the simulations by using a software framework called OpenMP which lets the user exploit the recent multi core processors from Intel or similar.

5.6.1 Traditional computing paradigm

Traditionally, software has been written for serial computation to be run on a single computer having a single Central Processing Unit (CPU). A problem is solved by an algorithm described by a discrete series of instructions, which are executed one after another - see Figure 5.7. The bottleneck is that only one instruction may execute at any moment in time and is radicated in the original Von Neumann architecture -see Figure 5.6 that was designed in the 1945. The control unit fetches instructions or data from memory, decodes the instructions and then sequentially coordinates operations to accomplish the programmed task.

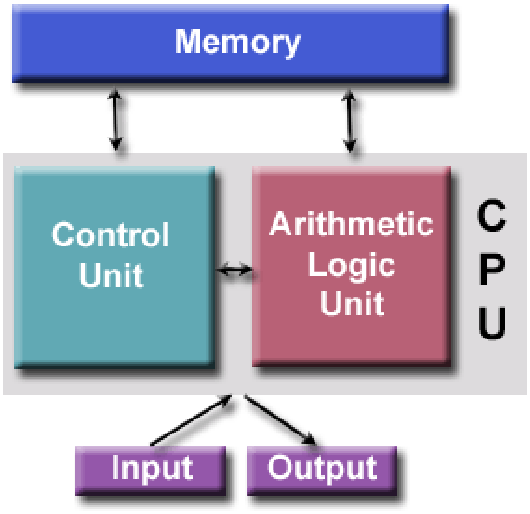


Figure 5.6: The von Neumann architecture is based on 4 components: the memory, the control unit, the arithmetic logic unit and an input and output channel.

Parallel computing adopts a new architecture to bypass the bottleneck of serial execution.

5.6.2 Parallel computing paradigm

Parallel computing is the simultaneous use of multiple computing resources to solve a computational problem. To avoid serial execution the algorithm is executed on multiple CPUs. To do so the problem must be broken into discrete parts that can be solved concurrently like in Figure 5.8 . Each part is further broken down to a series of instructions which are executed simultaneously on different CPUs.

The distribution of the computational load can be achieved by:

- A single computer with multiple processors

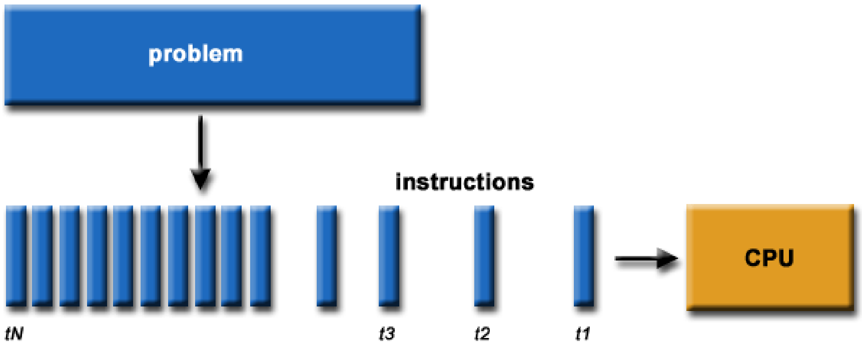


Figure 5.7: The problem is divided in a set of instructions which are executed at sequential time steps.

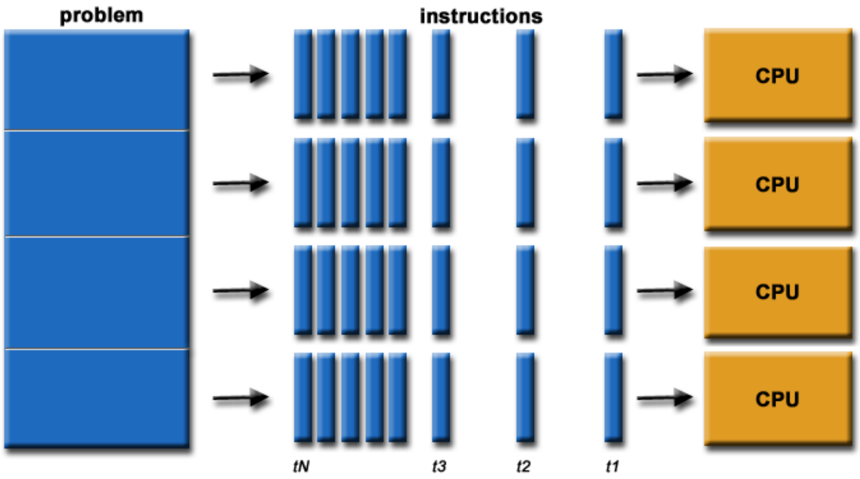


Figure 5.8: The problem is divided into blocks which are then divided in a set of instructions allocated to different CPUS.

- A computer network
- A combination of both

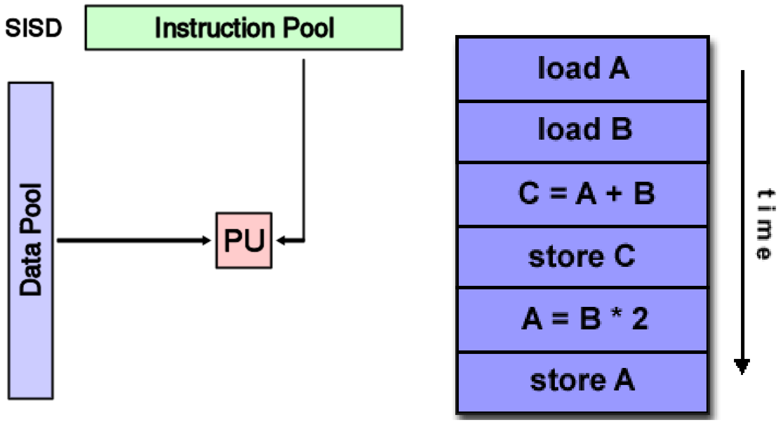
The problem needs to fulfil the basic property

- Broken apart into discrete pieces of work that can be solved simultaneously;
- Execute multiple program instructions at any moment in time;
- Solved in less time with multiple compute resources than with a single compute resource.

5.6.3 Flynn's Classical Taxonomy

Flynn's Taxonomy was introduced in 1966 to rationalise and classify the different type of parallel hardware architectures. The taxonomy contains 4 categories:

- Single Instruction Single Data: the CPU executes a single stream of instructions on a single stream of data like in Figure 5.9
- Single Instruction Multiple Data: each CPU executes the same stream of instructions on different streams of data like in Figure 5.10
- Multiple Instruction Single Data: each CPU executes different streams of instructions on the same stream of data like in Figure 5.11
- Multiple Instructions Multiple Data: each CPU executes different streams of instructions on the same stream of data like in Figure 5.12



```
load A
load B
C = A + B
store C
A = B * 2
store A
```

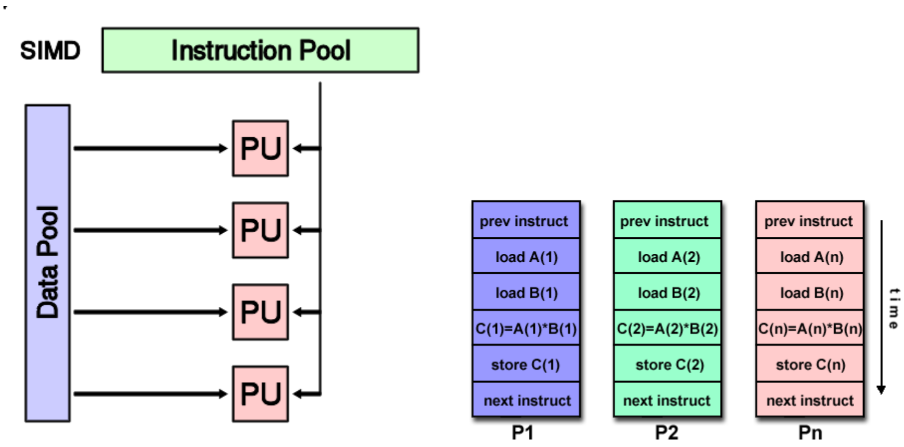
(a) SISD diagram

(b) SISD program

Figure 5.9:

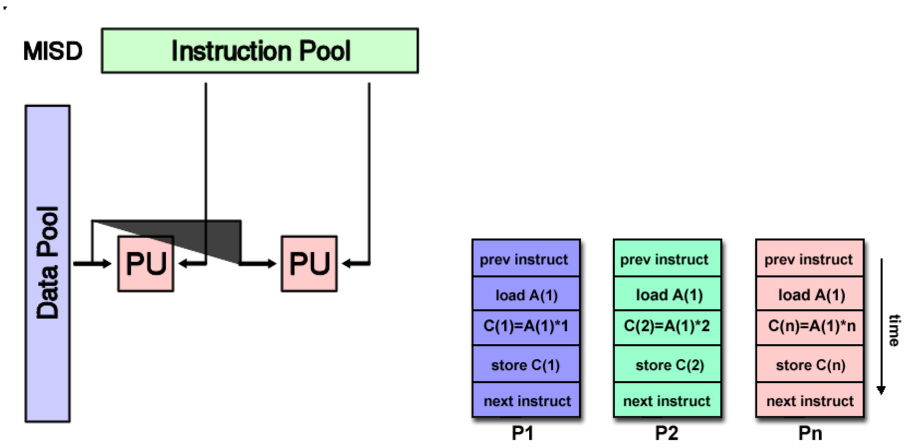
On top of this hardware layer, it was necessary to define a set of parallel programming models. The most common ones used are:

- Shared Memory
- Threads
- Message Passing



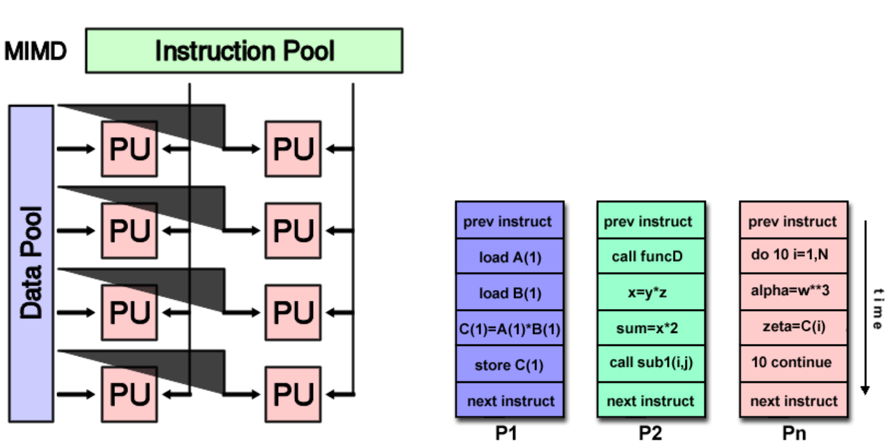
(a) SISD diagram (b) SISD program

Figure 5.10:



(a) SISD diagram (b) SISD program

Figure 5.11:



(a) SISD diagram

(b) SISD program

Figure 5.12:

- Data Parallel
- Hybrid

Those models are not dependent on any particular machine or memory architecture. There is no absolute best nor absolute worse, but surely there are better implementations or open source solutions.

One of the most successful model in the consumer level entry is is the Thread model, which is adopted both by Unix and Microsoft operative systems. In the threads model of parallel programming, a single process can have multiple, concurrent execution paths as in the Figure 5.13.

Threads are commonly associated with shared memory architectures and operating systems. The UNIX standard is POSIX and OpenMP.

5.6.4 Open MP

OpenMP stands for Open Multi-Processing but the original acronym was Open specifications for Multi-Processing via collaborative work between interested parties from the hardware and software industry, government and academia.

OpenMP is an application program Interface (API) that may be used to explicitly direct multi-threaded, shared memory parallelism. It is comprised of three primary API components (see Figure 5.14):

- Compiler Directives
- Runtime Library Routines
- Environment Variables

Open MP is Portable because the API is specified for $C/C++, Fortran$ and most major platforms have been implemented including Unix/Linux platforms and Windows NT Open MP is standardised and endorsed by a group of major computer hardware and software vendors The following table 5.1 compares the POSIX with the OpenMP standard.

The author decided to use OpenMP for implementing parallelism in the simulator because it is the most convenient choice in terms of language support and implementation.

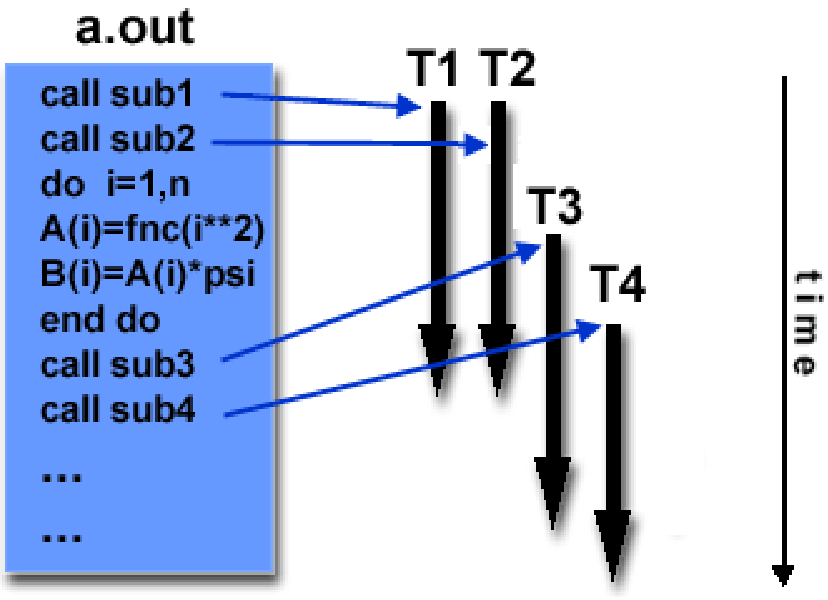


Figure 5.13: In the Thread Model a sequential program a.out can be executed as several threads T1,T2,T3,T4 which executes concurrently the sub routines of the program.

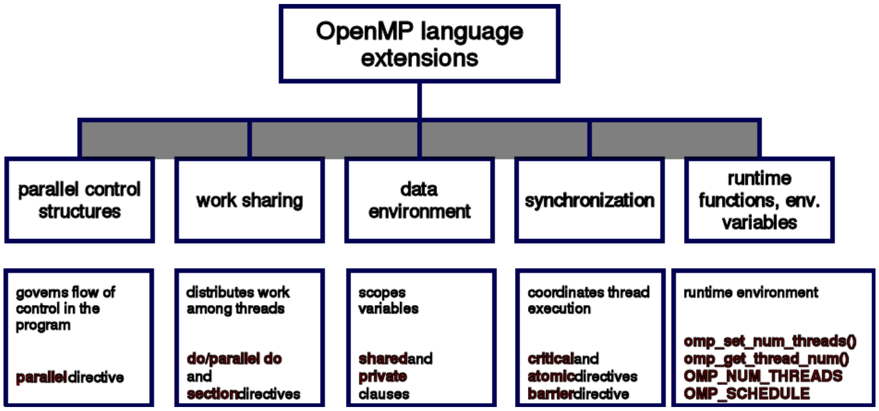

OpenMP language extensions
parallel control structures
work sharing
data environment
synchronization
runtime functions, env. variables
governs flow of control in the program
paralleldirective
distributes work among threads
do/parallel do and sectiondirectives
scopes variables
sharedand private clauses
coordinates thread execution
criticaland atomicdirectives barrierdirective
runtime environment
omp_set_num_threads()
omp_get_thread_num()
OMP_NUM_THREADS
OMP_SCHEDULE

Figure 5.14: OpenMP language structure: a set of pragma directives and keywords.

Standard	Posix	OpenMP
Language supported	C only	$C, C++$ and Fortran
Programming API	Library based requires parallel coding	Compiler directive based
Usability	Requires complex explicit coding	Easy and simple to use, incremental parallelism
Standard	IEEE POSIX 1003.1c standard (1995)	The OpenMP Fortran API was released October 28, 1997. The C/C++ API was released in late 1998.

Table 5.1: Table comparing the two most common standards

5.6.5 ABM simulation engine with OpenMp

The author developed an ABM engine called “Multi Parallel Agent” with codename “Empass” which is a minimalist particle engine written in $C \setminus C++$. The scope of this software project is to use OpenMP and thus have a fully portable parallel implementation. The motivation for developing a new simulator is that existing softwares did not do well in terms of language support, flexibility and scalability. The most popular engines for multi agent robot simulations are:

- RePast: A popular Java-based social complexity simulation toolkit.
- Ascape: Another popular Java-based social complexity simulation toolkit.
- Swarm: the venerable Objective-C and TCL-based social complexity simulator, from which RePast and Ascape (and MASON) owe much.
- TeamBots: A Java-based high-level, 2D abstract robotics simulator and hardware API.
- Player/Stage: A C++ based (but language-independent) 2D and 3D abstract robotics simulator and hardware API.
- Breve: A 3D simulation toolkit for MacOS X, Linux, and Windows using an interpreted language called Steve. Very impressive.
- StarLogo: A simulation toolkit in Logo, ostensibly for educational purposes, but extensible and powerful.
- NetLogo: Another, somewhat newer member of the Logo simulation family. Very nice!
- Processing A beautiful Java/OpenGL environment for simulation, animation, multimedia, and playing around.
- Enki: a fast 2D robot simulator in C++
- MASON: fast discrete-event multiagent simulation library core in Java, designed to be the foundation for large custom-purpose Java simulations

The most detailed simulators in terms of robot implementation are MASON and Player/Stage. NetLogo is similar in nature but the programming interface is based on a customised language grammar. Enki is minimalist and has a very fast implementation for collision detection and robot interaction. In terms of computational efficiency Player/Stage seems the best but is based on the Message Passing model (see Figure 5.15)for dividing the pure simulation details from the visualisation and control

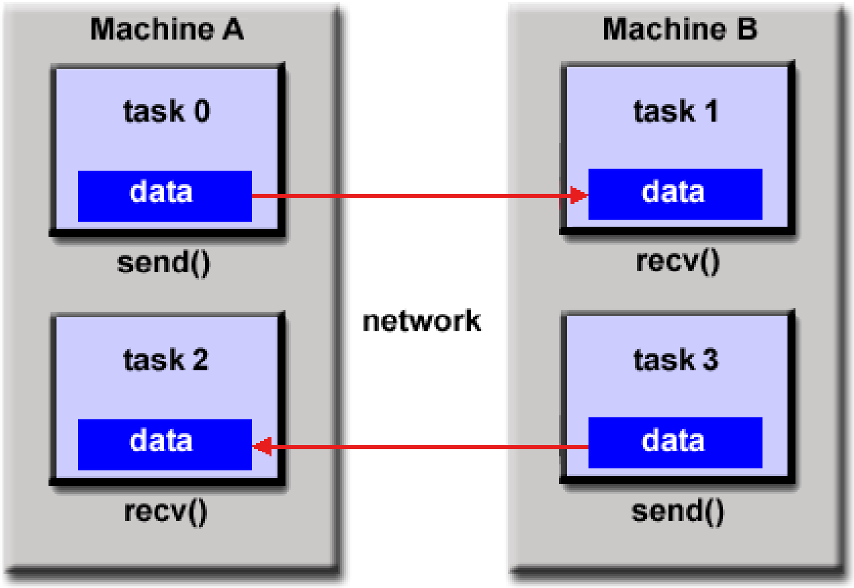


Figure 5.15: The message passing model can be implemented on the same machine or on several machines connected in the network. Each task has his own data and shares the intermediate results with messages across the network.

logic. All the Java based simulators trade off speed with compatibility because they are all based on the Java Virtual Machine which is good for portability but offers poor performances in execution time.

This design choice the author adopted was thus to develop a bare bones simulator which is Thread safe and can be used with OpenMP. The simulator Empass contains the following features:

- world environment can have a circular or square geometry
- robotic agent and obstacle entities modeled as disks
- particle physic computation for elastic and non elastic collisions
- efficient distance calculation with sparse matrixes in Boost library
- logging data in CSV format
- support for Gnuplot for visualizing data

The main use of this simulator is to run parametric simulations of the social system. For example in Figure 5.16 there are N different configurations of the social system where the black disks are the obstacles and the white disks are the agent. Each configuration will run for T_{sim} steps and is independent from the others.

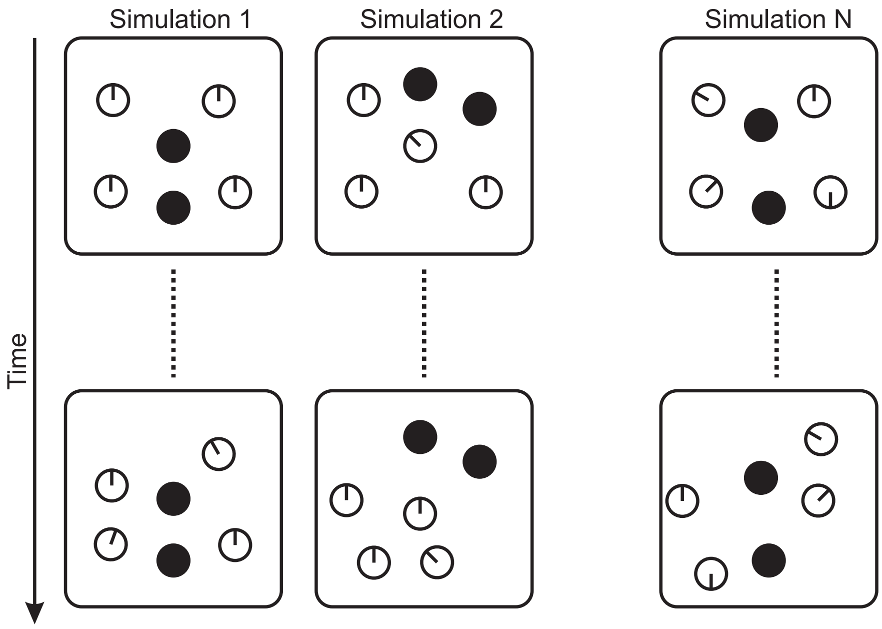


Figure 5.16: Graphical representation of a typical simulation with Empass. There are N simulations which differs for the initial configuration of the agents and obstacle location. Each simulation is independent from the others and thus can be parallelised.

The following code is a sequential implementation for a typical simulation run where there are 10 agents and 4 obstacles in a square world of 100 for 100 units. There are a total of $Nsim = 100$ different simulations that have to be run for $T_{sim} = 10000$ time steps. Each configuration will generate different results because the initial conditions are randomised.

```
N=10;        // agents
```

```
M=4;        // obstacles
simtime=10000;   // simulation time

for(int conf=0;conf<Nsim;conf++)
{
    //create 100x100 unit square worlds
    world[conf]=World2DCartesian(100,100);
}

for(int conf=0;conf<Nsim;conf++)
{

    world[conf].run(N,M,simtime,false);

}
```

The previous code is the implementation of the same simulation code but using the **parallel region construct** and the **work sharing construct** as in Figure 5.17.

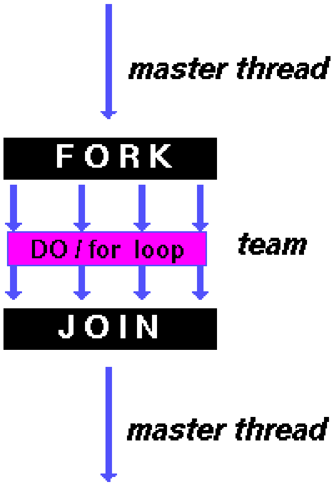


Figure 5.17: How OpenMP fork and join a while loop.

```
N=10;        // agents
M=4;        // obstacles
simtime=10000;   // simulation time
int chunk=10;
int conf=0;

for(int conf=0;conf<Nsim;conf++)
{
    //a 100x100 unit square world
    world[conf]=World2DCartesian(100,100);
}

#pragma omp parallel shared(N,M,simtime,world,chunk) private(conf)
{
    #pragma omp for schedule(dynamic,chunk) nowait
    for(conf=0;conf<Nsim;conf++)
    {
       world[conf].run(N,M,simtime,false);
    }
}
```

The tests were performed on two Linux machines, one with an Intel Core2 Quad CPU Q6600 at 2.40GHz with 4GB RAM DDRII memory and another one with Intel Core Duo at 2.40 GHz with 2 GB RAM DDR memory. The Table 5.2 shows how the execution time is drastically reduced by a factor of 3.25, ideally one could achieve a factor of 4.0 because there are virtually 4 CPU but this is not possible due to the actual hardware implementation. For $N_{sim} = 10000$ the experimenter could easily have waited 1 minutes and 13 seconds whereas with the parallel implementation only 35.63 seconds. It is also interesting to note that there was not any significant difference in terms of performance between the Intel Core Duo and the Intel Quad CPU.

Table 5.2: Table with some benchmarking measures

Configuration	Processor	Sequential	OpenMP
$N_{sim} = 1000$	Intel Quad CPU	0m13.285s	0m4.082s
$N_{sim} = 10000$	Intel Quad CPU	1m13.178s	0m35.637s
$N_{sim} = 1000$	Intel Core Duo CPU	0m7.088s	0m3.584s
$N_{sim} = 10000$	Intel Core Duo CPU	1m10.840s	0m35.647s

This is only a very simple optimisation included in the simulator but there are other possible optimisation strategies for the collision computation or the communication protocol. The next section describes a very important optimisation which was not implemented for time constraints but is very important for future simulators.

5.6.6 Pruning the parallel simulator

Another optimisation that can be introduced is to “prune” the task execution. In some occasions for a particular set of parameters a simulation can reach a a looping stage where the agents are repeating their trajectories due to an unfortunate positioning of the obstacles. In this condition the simulation can be terminated because , the data produced is of no use. In Figure 5.18, the simulation $X + 1$ must be terminated because the agent has incurred in a trajectory loop due to the obstacle configuration. Simulation X instead does not contain any loop condition. If the detection of the loop is shared between the simulation tasks this will improve even more dramatically the performances by essentially removing the tasks which are not useful. This feature was not implemented as it required a more advanced inter communication between tasks and is left as a future function. There is also a novel approach that can be used to predict if an initial configuration of agents will reach a loop condition and is described in Chapter 4.2.1.

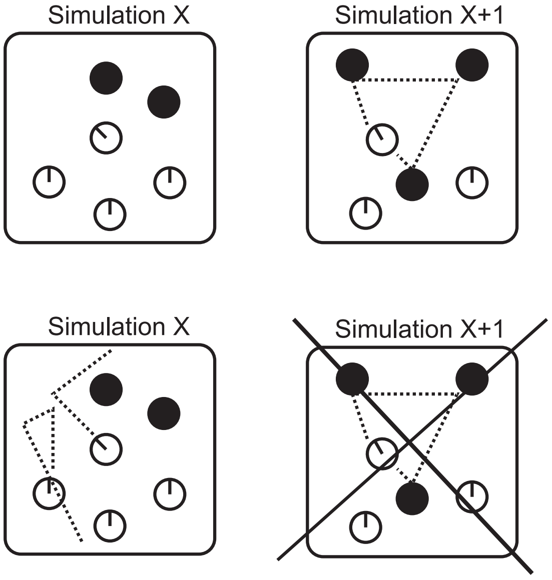


Figure 5.18: The agent in the simulator labelled with $X + 1$ incurred in a trajectory loop, and thus the simulation is stopped.

5.6.7 Discussion

The importance of supporting the Thread model in computer simulation is becoming of more importance within the consumer desktop industry. Intel multi core technology is a clear example, the Core 2 Duo CPU was released on July 2006 and the latest version Core i7 EE with 6 cores was release in January 2011. This means that more and more researcher have access to multi core hardware but do not have the tools to speed up their simulations. There is also another interesting growing branch of optimisation based on the use of the CUDA SDK based on the Nvidia GPU hardware. For example, CUDA now accelerates AMBER, a molecular dynamics simulation program used by more than 60,000 researchers in academia and pharmaceutical companies worldwide to accelerate new drug discovery. It seems that computing is evolving from "central processing" on the CPU to "co-processing" on the CPU and GPU. To enable this new computing paradigm, NVIDIA invented the CUDA parallel computing architecture that is now shipping in GeForce, ION, Quadro, and Tesla GPUs, representing a significant installed base for application developers. The company producing AMD and ATI hardware are also trying to catch up with this field but they are still behind a useful implementation. This is certainly an interesting trend for future researchers as well as videogame producers.

5.7 Implementation of a swarm system

There are several reasons to realise a swarm system:

1. proof of concept
2. technology showcase during open events
3. solve a business task

Software simulations don not take into account the complexity of the real world and uncertainty introduced by mechatronic systems, therefore a real implementation is likely to spot unexpected behaviour of the system. A live swarm system can be used to sponsor research and attract new students/researchers in this area and is also fun to watch. Additionally it can be extended to solve real business tasks, for example the company "Kiva Systems" used a swarm system to improve the performance of order fulfilment [2]. Nevertheless there are 3 main challenges:

- scaling down the costs for large systems
- operating conditions
- replicability

A swarm systems is likely composed of many agents, otherwise it doesn't make sense to call it a swarm: therefore the price of every single unit must be very low. Hence the unit must have different hardware according to the task it was designed for: if the task is to move boxes we need grip actuators to move them and cameras to determine their location. I designed my system to be easy to replicate on the hardware and software side, this is important for other researchers if they want to test new algorithms or replicate the results. The robots/agents are positioned on a rectangular playground with a white background, a food patch is painted in the centre as a graded circle. The robots must have reflective sensors pointing down to detect food signals, contact sensors to avoid each other as well as walls and a communication method to exchange information.

5.7.1 The playground

The playground is printed on a grey scale A3 paper. The playground dimensions are reported in the figure. The robot we are using, called 3pi, from Pololu has the same diameter has the inner black spot. The background and the wall are at high contrast white/black and the blob at the centre has a gradient texture: the robot will pass over it to calibrate the minimum, maximum and average readings from its reflective sensors looking down to the floor.

5.7.2 Hardware platform and software development

There were two main choices about the hardware implementation:

- custom design: designing the electronic and mechanical part
  - advantages: total control on choice of processor, sensors, actuators, communication
  - disadvantages: manufacturing can become an issue: assembly errors, testing and labour cost. Not easy to be used by external researchers, documentation and materials must be provided but mechanical parts are easy to find etc...
- open design:
  - advantages: available virtually to everybody, easy to use and to obtain
  - disadvantages: limited control on the hardware

[2] The company website is: http://www.kivasystems.com/index.html

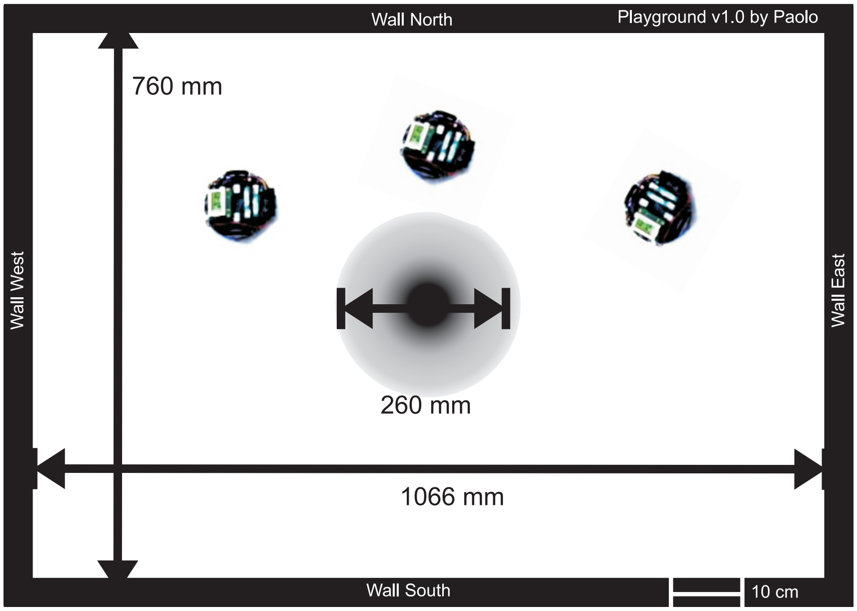


Figure 5.19: The playground dimensions

According to the table above I decided to use 2 commercial popular robots so that virtually everybody can use them and replicate my results. The first choice is the lego mindstorm nxt which was released by Lego as a fully open source documented platform on 1 May 2006. Lego sold 150,000 units in 2007 worldwide. The second choice is the 3pi from Pololu (see specifications here 5.7.8). I have chosen these two because they are in a different price range (150 and $60 respectively) and have a different hardware complexity. To simplify the software development I:

1. developed a common C++ library called UICO
2. compiled, tested and debugged under x86 (a common desktop pc)
3. adapted to the programming environments of the lego nxt and pololu 3pi
4. verified again the program behaviour in the real application

All the libraries I developed are available on-line and can be easily installed and compiled by any other user.

5.7.3 Lego NXT mindstorm

Lego has 10 years experience in producing educational robot kits, NXT[3] is an updated version of the RCX[4] kit. With this product Lego has released full hardware and software specification of the kit[5] so that users can develop their own firmware and interface different hardware NXT controller. Recently Professor Masaaki Mizuno (Department of Computing and Information Sciences, Kansas State University) did a port of the TOPPERS/ATK to the NXT naming it **nxtOSEK**. **nxtOSEK** [6]

[3] http://mindstorms.lego.com/

[4] http://www.lego.com/eng/education/mindstorms/home.asp?pagename=rcx

[5] http://mindstorms.lego.com/eng/Overview/nxtreme.aspx

[6] http://lejos-osek.sourceforge.net/

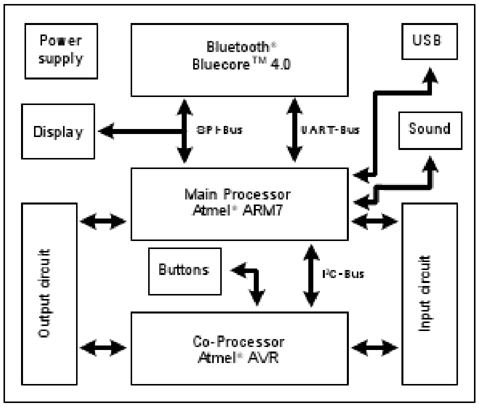


Figure 5.20: Hardware block diagram of the NXT brick. For a better description see Appendix 5.7.7

consists of device driver of leJOS NXJ C/Assembly source code, TOPPERS/ATK (Automotive Kernel, formerly known as TOPPERS/OSEK) and TOPPERS/JSP Real-Time Operating System source code that includes ARM7 (ATMEL AT91SAM7S256) specific porting part, and glue code to make them work together. nxtOSEK can provide:

- ANSI C/C++ programming environment by using GCC tool chain
- C API for NXT Sensors, Motor, and other devices
- C++ API for NXT Sensors and Motor which include many third party sensors
- TOPPERS/ATK provided real-time multi tasking features proven in automotive industry
- TOPPERS/JSP provided real-time multi tasking features that complied with Japan original open RTOS specification $\mu ITRON 4.0$
- Fast execution and less memory consumption
- There are three ways to upload the nxtOSEK application to the NXT
  1. Using John Hansen's Enhanced NXT firmware (multiple nxtOSEK programs can be uploaded to a NXT. However, a nxtOSEK program has to be less than 64Kbytes)
  2. Using NXT BIOS (max. 224Kbytes single nxtOSEK program uploaded to Flash)
  3. Direct boot from RAM (max. 64Kbytes single nxtOSEK program uploaded to RAM, no Flash write)
- Many examples (including NXTway-GS and NXT GT...)

The robot has been built using lego technic pieces from the kit: it has 2 motors with rubber wheels and a casting wheel on the back to allow differential steering. Two reflective sensors are placed on the bottom to detect food sources. Two contact sensors are placed on the front so that the robot can avoid walls and other robots.

The control program of the robot is using the already mentioned UICO library, and is composed of 4 tasks. They are scheduled using rate-monotonic full-preemptive scheduling (RMS): a scheduling algorithm used in real-time operating systems with a static-priority scheduling class. The static priorities are assigned on the basis of the cycle duration of the job: the shorter the cycle duration is, the

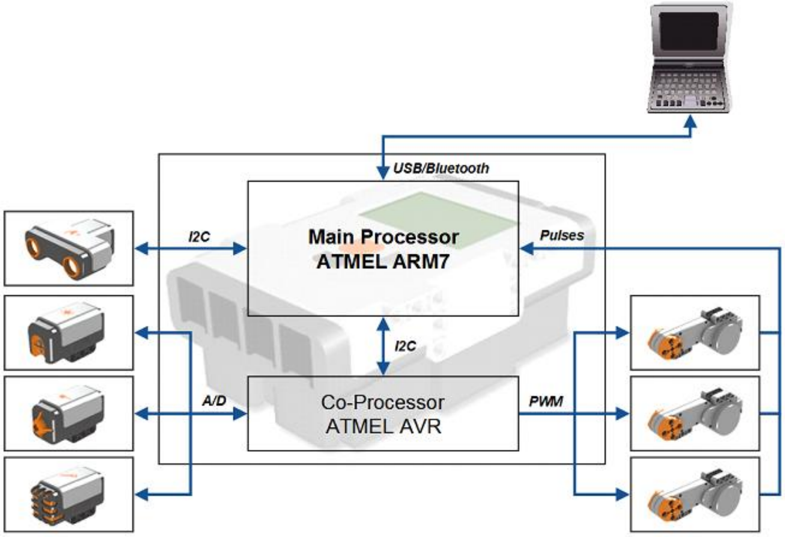
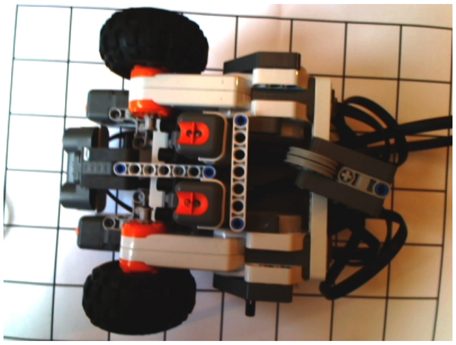


Figure 5.21: This figure tells that communication between the main ARM7 processor (ATMEL AT91SAM7S256) and Sensors/Servo Motors is done via the co-processor (ATMEL AVR) except for the Ultrasonic Sensor and acquisition of Servo Motor revolutions. For nxtOSEK, the most important factor to access Sensors/Servo Motors is the communication with the co-processor via I2C serial bus. This system architecture definitely influences the software run-time environment of nxtOSEK. The main ARM7 processor accesses Sensors (to read sensor A/D value) and Servo Motors (to set PWM duty ratio and break mode) independently every 2 msec through a 1 msec periodical Interrupt Service Routine (ISR) of LEJOS NXJ platform. Servo Motors revolutions are directly captured by pulse triggered ISRs of LEJOS NXJ platform. Ultrasonic Sensor has its brain directly communicate with the main ARM7 processor via another I2C communication channel. TOPPERS ATK is similar to the following version of OSEK OS/OIL according to the TOPPERS project. nxtOSEK restricts several TOPPERS ATK features due to the system architecture. User should not use ISR definitions and Interrupt handling API [8].



(a) LegoRobot bottom view: left and right light sensors

higher the job priority. In the diagram the higher priority 4 is assigned to the Control task with a 10 mseconds period, then in ascending order priority 3 to the Avoid Task, priority 2 to the DataLog task and priority 1 to the LCD task.

5.7.4 Toppers

TOPPERS[9] is an acronym for Toyohashi OPen Platform for Embedded Real-time Systems. Toyohashi is a city located in Japan, and the name was selected based on project leader Professor Takadas association with Toyohashi University of Technology when the project was started. It is based on Open Source and Free Software and thus easy to port on any embedded platform.

5.7.5 OSEK

OSEK/VDX is a joint project of the automotive industry. It aims at an industry standard for an open-ended architecture for distributed control units in vehicles. The specification of the OSEK operating system represents a uniform environment which supports efficient utilisation of resources for automotive control unit application software. The OSEK operating system is a single processor operating system meant for distributed embedded control units. One of the goals of OSEK is to support the portability and re-usability of application software. Therefore the interface between the application software and the operation system is defined by standardised system services with well-defined functionality. Use of standardised system services reduces the effort to maintain and to port application software and development cost. This is why it was so easy to port OSEK onto the Lego NXT plaftorm. The OSEK operating system serves as a basis for application programs which are independent of each other, and provide their environment on a processor. The OSEK operating system enables a controlled real-time execution of several processes which appear to run in parallel. The OSEK operating system provides a defined set of interfaces for the user. These interfaces are used by entities which are competing for the CPU. There are two types of entities:

1. Interrupt service routines managed by the operating system.
2. Tasks (basic tasks and extended tasks).

The hardware resources of a control unit can be managed by operating system services. These operating system services are called by a unique interface, either by the application program or internally within the operating system. OSEK defines three processing levels:

[9] http://www.toppers.jp/en/index.html

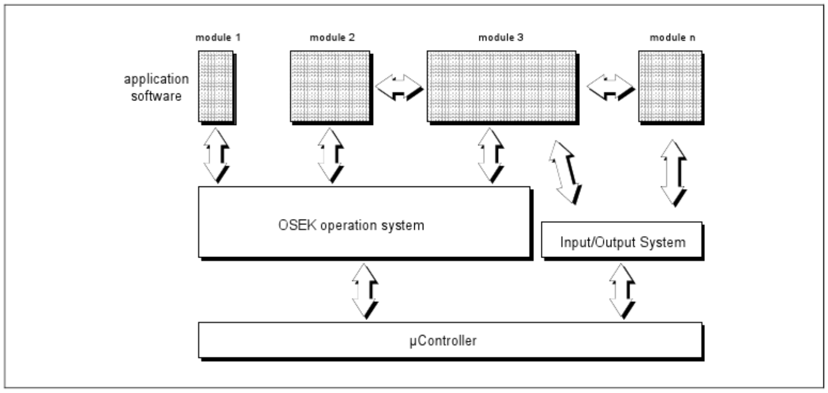


Figure 5.22: Software interfaces inside ECU

- Interrupt level
- Logical level for scheduler
- Task level

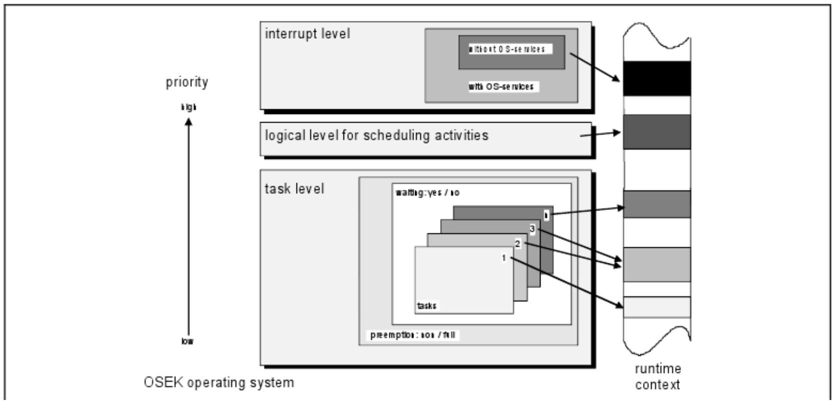


Figure 5.23: Processing levels of the OSEK operating system

For better portability of application software, the OSEK defines a language for a standardised configuration information. This language "OIL" (OSEK Implementation Language) supports a portable description of all OSEK specific objects such as "tasks" and "alarms" etc. Website: http://www.osek-vdx.org/

5.7.6 OIL

To reach the OSEK goal of portable software, a way has been defined to describe the configuration of an application using OSEK. This specification only addresses a single central processing unit (CPU) in an electronic control unit (ECU[10]), not an ECU network. The goal of OIL is to provide a mechanism

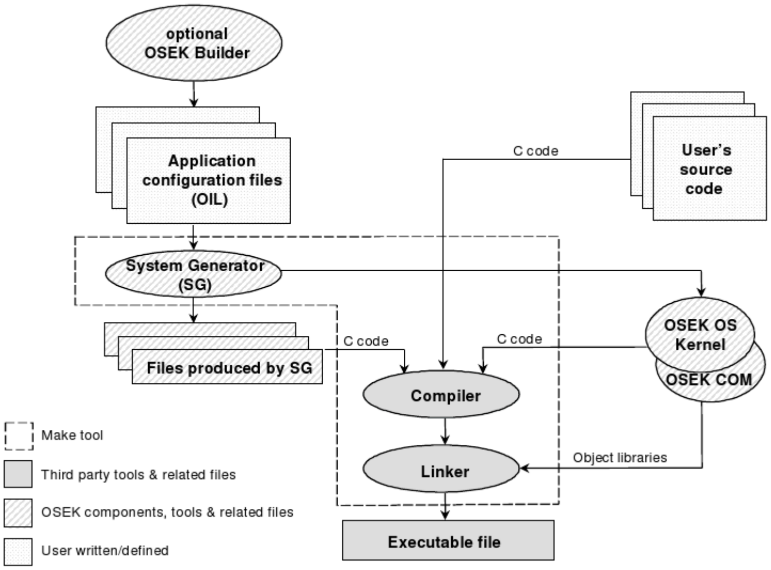


Figure 5.24: Example of development process for applications

to configure an OSEK application inside a particular CPU. This means for each CPU there is one OIL description. All OSEK system objects are described using OIL objects. The OIL description of the OSEK application is considered to be composed of a set of OIL objects. A CPU is a container for these OIL objects. OIL defines standard types for its objects. Each object is described by a set of attributes and references. OIL defines explicitly all standard attributes for each OIL object. Each OSEK implementation, like nxtOsek, can define additional implementation-specific attributes and references. The OIL configuration I'm using in my robots:

```
#include ”implementation.oil”

CPU ATMEL_AT91SAM7S256
{
  OS LEJOS_OSEK
  {
    STATUS = EXTENDED;
    STARTUPHOOK = FALSE;
    ERRORHOOK = FALSE;
    SHUTDOWNHOOK = FALSE;
    PRETASKHOOK = FALSE;
    POSTTASKHOOK = FALSE;
    USEGETSERVICEID = FALSE;
    USEPARAMETERACCESS = FALSE;
    USERESSCHEDULER = FALSE;
```

[10] Engine Control Unit

```
};

/* Definition of application mode */
APPMODE appmode1{};

EVENT SensorEventMask {
  MASK = AUTO;
};

EVENT SleepEventMask {
  MASK = AUTO;
};


/* Show status information */
TASK TaskLCD
{
        AUTOSTART = TRUE {
        APPMODE = appmode1;
        };
  EVENT = SensorEventMask;
  EVENT = SleepEventMask;
  PRIORITY = 1; /* Smaller value means lower priority */
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512; /* Stack size */
};

 TASK TaskDataLog
{
        AUTOSTART = TRUE {
        APPMODE = appmode1;
        };
  EVENT = SensorEventMask;
  EVENT = SleepEventMask;
  PRIORITY = 2; /* Smaller value means lower priority */
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512; /* Stack size */
};

 TASK TaskAvoid
{
        AUTOSTART = TRUE {
        APPMODE = appmode1;
        };
  EVENT = SensorEventMask;
  EVENT = SleepEventMask;
  PRIORITY = 3; /* Smaller value means lower priority */
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512; /* Stack size */
};

/* Reak time task with the digital controller */
TASK TaskControl
{
      AUTOSTART = TRUE {
      APPMODE = appmode1;
       };
  EVENT = SensorEventMask;
  EVENT = SleepEventMask;
  PRIORITY = 4; /* Smaller value means lower priority */
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512; /* Stack size */
};



TASK SensorMonitorTask {
  AUTOSTART = FALSE;
  PRIORITY = 1;
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512;
};

/* Definition of OSEK Alarm Counter */
COUNTER SensorMonitorCounter
{
  MINCYCLE = 1;
  MAXALLOWEDVALUE = 10000;
  TICKSPERBASE = 1; /* One tick is equal to 1msec */
};

/* Definition of SensorMonitorTask execution timing */
ALARM cyclic_alarm
{
  COUNTER =  SensorMonitorCounter;
  ACTION = ACTIVATETASK
  {
      TASK = SensorMonitorTask;
  };
  AUTOSTART = TRUE
  {
      ALARMTIME = 1;
      CYCLETIME = 10; /* Task is executed every 10msec */
      APPMODE = appmode1;
  };
```

```
};

  /* Definition of TaskLCD execution timing */
ALARM cyclic_alarmLCD
{
  COUNTER =  SensorMonitorCounter;
  ACTION = ACTIVATETASK
  {
      TASK = TaskLCD;
  };
  AUTOSTART = TRUE
  {
      ALARMTIME = 1;
      CYCLETIME = 500; /* LCD display is updated every 500 msec */
      APPMODE = appmode1;
  };
};

/* Definition of TaskDataLog execution timing */
ALARM cyclic_alarmDataLog
{
  COUNTER =  SensorMonitorCounter;
  ACTION = ACTIVATETASK
  {
      TASK = TaskDataLog;
  };
  AUTOSTART = TRUE
  {
      ALARMTIME = 1;
      CYCLETIME = 100; /* Data is logged every 100 msec */
      APPMODE = appmode1;
  };
};

 /* Definition of TaskAvoid execution timing */
ALARM cyclic_alarmAvoid
{
  COUNTER =  SensorMonitorCounter;
  ACTION = ACTIVATETASK
  {
      TASK = TaskAvoid;
  };
  AUTOSTART = TRUE
  {
      ALARMTIME = 1;
      CYCLETIME = 8; /* Avoiding task is executed every 8 msec */
      APPMODE = appmode1;
  };
};
```

```
      /* Definition of TaskControl execution timing */
  ALARM cyclic_alarmControl
  {
    COUNTER =  SensorMonitorCounter;
    ACTION = ACTIVATETASK
    {
        TASK = TaskControl;
    };
    AUTOSTART = TRUE
    {
        ALARMTIME = 1;
        CYCLETIME = 10; /* Neural control is sampled at 10 msec */
        APPMODE = appmode1;
    };
  };


};
```

5.7.7 LegoRobot

Here is a summary list of hardware specifications for the NXT brick:

- Main processor: Atmel 32-bit ARM processor, AT91SAM7S256, 256 KB FLASH, 64 KB RAM.
- Bluetooth: CSR BlueCoreTM 4 v2.0 +EDR System
- USB 2.0 communication: full speed port (12 Mbit/s)
- 4 input ports: 6-wire interface supporting both digital and analog interface and 1 high speed port, IEC 61158 Type 4/EN 50170 compliant
- 3 output ports: 6-wire interface supporting input from encoders
- Display: 100 x 64 pixel LCD black. white graphical display
- Loudspeaker: sound output channel with 8-bit resolution, supporting a sample rate of 2-16 KHz
- user interface: 4 rubber buttons
- power source: batteries or usb

Task functions

The Task Control is the most important one: is composed by a 5 states machine.

- Init: initialisation of the software library and hardware controller
- Calibration: the robot crosses the food patch in the playground to calibrate its sensors. It calculates minimum, maximum and average reflectance of the ground. These values will be used to scale the proximal and distal food signals.
- Reflex: the agent is purely reactive.
- Learning: the agent is learning to associate the distal signal to the proximal stimulus.

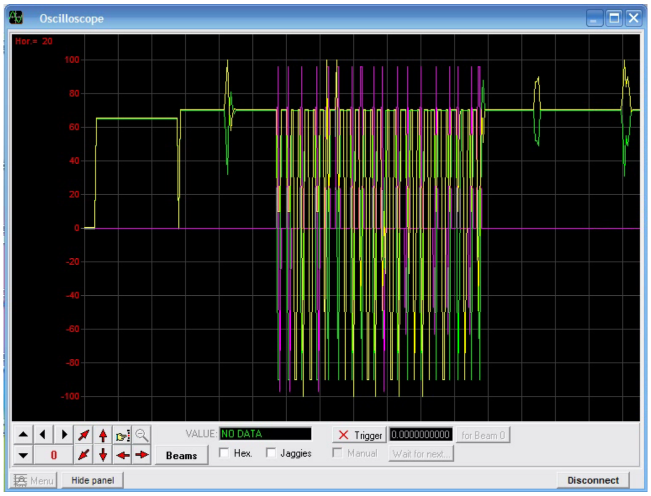
The Task Avoid is responsible for the avoidance task: the contact sensors are connected to 2 IIR filters which produce a delayed back-turning motor response. It's also responsible to produce a tone relative to the energy of the robot. The Task Data Log is used for 2 purposes:

- to upload the signals of the robot to a user desktop via the bluetooth SPP (serial port protocol) interface
- to communicate via bluetooth with the other robots

The data logging function was crucial for debugging the robot and will be used to compute the information measure for every agent. On the Desktop side, a simple program written in C++ - at the moment only for windows, is available on the website - it reads the data from the serial COM port and visualises it. There are 2 programs doing it:

- one operates in off-line mode: it saves all the incoming data in the computer in a file. This file -in CSV format- is then importend in Matlab to plot the relative signals
- one operates in on-line mode: it receives the data and plot it in almost real time in an oscilloscope like manner

Both programs are available on the website to download (see Appendix D).



VALUE: NO DATA

Trigger 0.0000000000

Figure 5.25: The bluetooth logger developed for windows. It has an interface similar to an oscilloscope. It logs the signal of interest. In this case the energy and the motor outputs.

How to guarantee real time operations?

In Osek the task model is composed by 4 states as described in Fig.5.26.

A rate monotonic analysis is necessary to guarantee that for a particular application every task will be executed and completed in time. In Fig.5.27 there's a simple explanation about how it works:

- after the system boot all tasks are in ready state
- the task 4 with higher priority and is put in running state
- task 4 is terminated, the task with lower priority, 3, is put in running state

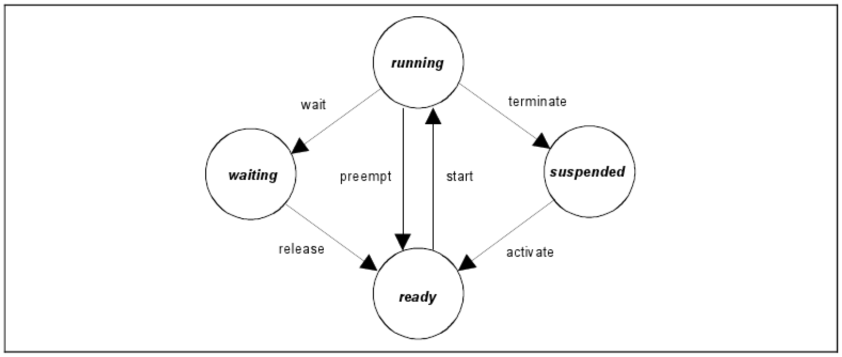


Figure 5.26: A task must be able to change between several states, as the processor can only execute one instruction of a task at any time, while several tasks may be competing for the processor at the same time. The OSEK operating system is responsible for saving and restoring task context in conjunction with task state transitions whenever necessary.

- after 10 ms task 4 is put again in run mode
- task 4 is terminated and task 2 is put in run mode
- task 1 is finally in run mode after task 2 is terminated
- nevertheless because task 4 with higher priority is due at 20 msec, task 2 is pre-empted (waiting state) in favour of task 4
- task 4 terminates and task 1 finishes etc.

Rate monotonic scheduling considers the 4 threads in the system and determines how much time is needed to meet the guarantees for the set of threads in question. It assumes that:

- No resource sharing (processes do not share resources, e.g. a hardware resource, a queue, or any kind of semaphore blocking or non-blocking (busy-waits))
- Deterministic deadlines are exactly equal to periods
- Static priorities (the task with the highest static priority that is runnable immediately preempts all other tasks)
- Static priorities assigned according to the rate monotonic conventions (tasks with shorter periods/deadlines are given higher priorities)
- Context switch times and other thread operations are free and have no impact on the model

Liu and Layland (1973) proved that for a set of n periodic tasks with unique periods, a feasible schedule that will always meet deadlines exists if the CPU utilization is below a specific bound (depending on the number of tasks). The schedulability test is:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(\sqrt[n]{(2)} - 1). \tag{5.29}$$

where C_i is the computation time, and T_i is the release period (with deadline one period later). For example $U = 0.8284$ for $n = 2$. When the number of processes tends towards infinity this expression will tend towards:

$$\lim_{n \to \infty} n(\sqrt[n]{(2)} - 1) = \ln(2) \approx 0.693147 \tag{5.30}$$

So a rough estimate is that RMS in the general case can meet all the deadlines if CPU utilization is 69.3%. The other 30.7% of the CPU can be dedicated to lower-priority non real-time tasks. It is known that a randomly generated periodic task system will meet all deadlines when the utilization is 85% or less, however this fact depends on knowing the exact task statistics (periods, deadlines) which cannot be guaranteed for all task sets. In my implementation in order to calculate U, $C_{1,2,3,4}$ must be computed using the Timer of the controller: execute every singular task individually and time it.

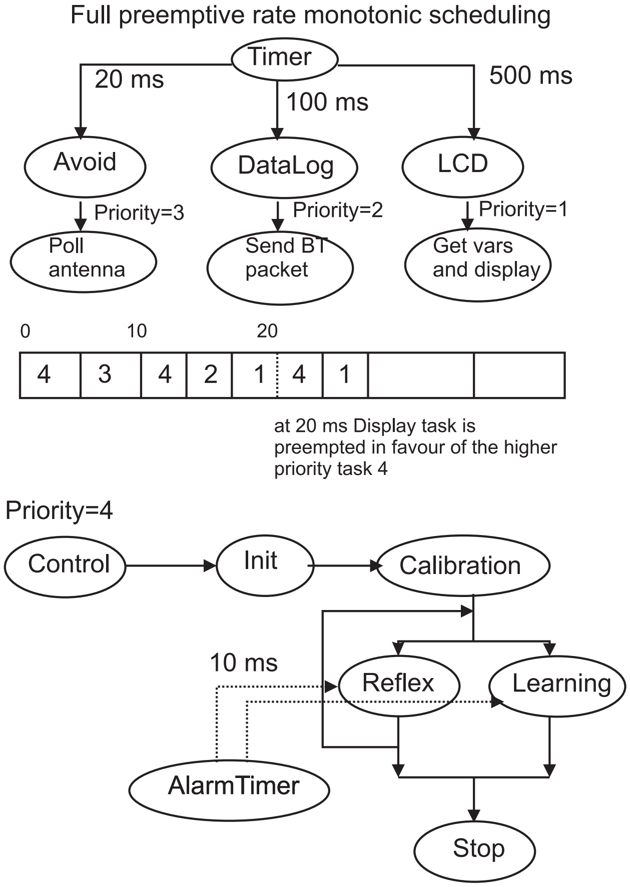


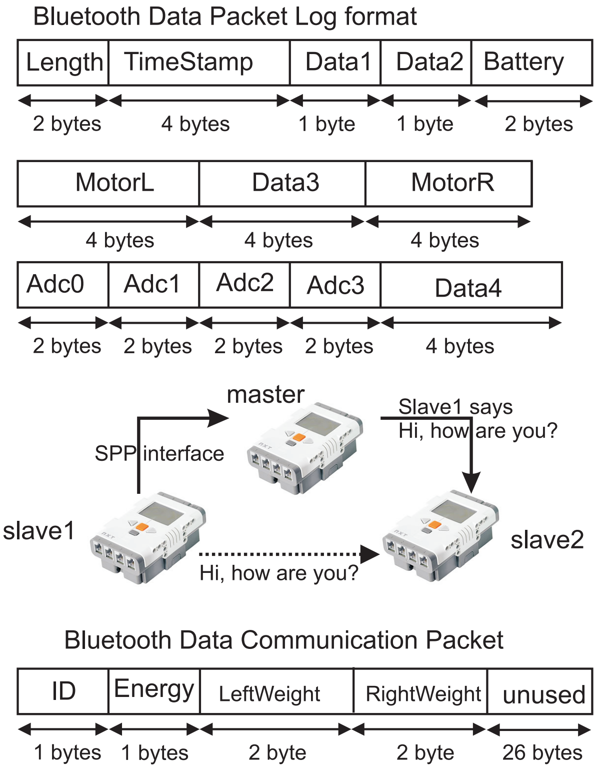
Figure 5.27: The control program task diagram. There are 4 tasks scheduled in full-preemption mode.

Communication and identification

The most important aspect in our experiment is communication and identification. We would like test 2 types of communication:

- 1 to many: every agent broadcasts its state to everybody else
- 1 to 1: every agent communicates its state to a close peer

Communication is achieved by using the BluetoothCore and is structured as master-slave: in a swarm slaves sends their phonemes (language primitives) to the master which has the role to dispatch them to the addressed receivers. Every robot is identified by an ID which is inserted in every communication data packet. The packet is 32 byte long and its structure is in Fig.5.28. In this simple setup every robot is communicating its ID energy level to others. The robot also communicates his energy level by producing a sound so that a human observer can have feedback about what's going on. The lego speaker can play a tone, given its frequency, duration and volume. Frequency is audible from about 31 to 2100 Hertz and different frequencies are associated to different robots. The duration is in hundreds of a seconds (centiseconds, not milliseconds) and is truncated at 256, so the maximum duration of a tone is 2.56 seconds. The volume of the tone is proportional to the robot's energy which is normalised from 0 to 100 (internally is a float from 0.0 to 1.0).



Bluetooth Data Packet Log format

Length	TimeStamp	Data1	Data2	Battery
2 bytes	4 bytes	1 byte	1 byte	2 bytes

MotorL	Data3	MotorR
4 bytes	4 bytes	4 bytes

Adc0	Adc1	Adc2	Adc3	Data4
2 bytes	2 bytes	2 bytes	2 bytes	4 bytes

Bluetooth Data Communication Packet

ID	Energy	LeftWeight	RightWeight	unused
1 bytes	1 bytes	2 byte	2 byte	26 bytes

Figure 5.28: The bluetooth data logging structure.

5.7.8 Pololu

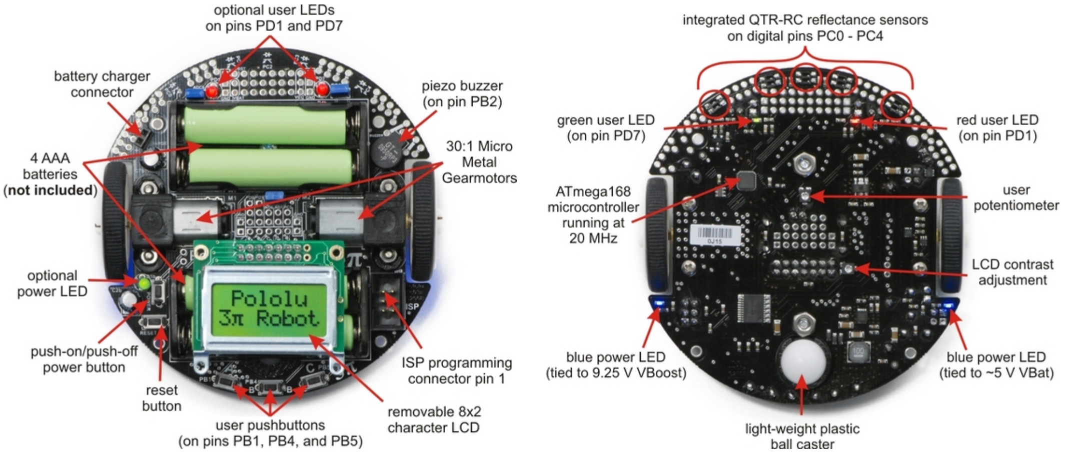
The Pololu 3pi 5.7.8 robot is a complete, high-performance mobile platform featuring two micro metal gearmotors, five reflectance sensors, an $8x2$ character LCD, a buzzer, and three user pushbuttons, all connected to a C-programmable ATmega168 microcontroller. Capable of speeds exceeding 3 feet per second ($100cm/second$), 3pi is a great first robot for ambitious beginners and a perfect second robot for those looking to move up from non-programmable or slower beginner robots. **Dimensions:** Size:

9.5cm/3.7” diameter, **Weight:** 83g/2.9oz without batteries **General specifications**

- Processor: $ATmega168@20MHz$
- Motor driver: TB6612FNG
- Motor channels: 2
- User I/O lines: 21
- Minimum operating voltage: 3 V2
- Maximum operating voltage: 7 V2
- Maximum PWM frequency: 80 kHz
- Flash program memory: 16KB
- Extra 512 bytes of persistent flash memory[11]
- Data memory: 1KB
- Reverse voltage protection?: Y
- External programmer required?: Y
-

Notes:

1. Digital I/O lines PD0 and PD1 are available; two more analog inputs and one analog/digital pin can be made available by removing jumpers and disabling special features of the board.
2. Designed for use with 4 x AAA NiMH or Alkaline cells. A step-up regulator boosts the motor voltage to 9.25 V.


optional user LEDs on pins PD1 and PD7
battery charger connector
piezo buzzer (on pin PB2)
4 AAA batteries (not included)
30:1 Micro Metal Gearmotors
optional power LED
Pololu 3π Robot
push-on/push-off power button
reset button
ISP programming connector pin 1
user pushbuttons (on pins PB1, PB4, and PB5)
removable 8x2 character LCD
integrated QTR-RC reflectance sensors on digital pins PC0 - PC4
green user LED (on pin PD7)
red user LED (on pin PD1)
ATmega168 microcontroller running at 20 MHz
user potentiometer
LCD contrast adjustment
blue power LED (tied to 9.25 V VBoost)
blue power LED (tied to ~5 V VBat)
light-weight plastic ball caster

(a) Pololu robot bottom view: left and right light sensors

(b) Pololu robot up view: left and right contact senros and LCD display

Figure 5.29: Pololu robot up and bottom view

The processor is programmed using an external AVR ISP programmer such as the Orangutan USB programmer. The popular, free GNU C/C++ compiler works perfectly with the 3pi, Atmels AVR

[11] is provided on the microcontroller for data logging or long-term learning applications

Studio provides a comfortable development environment, and an extensive set of libraries provided by Pololu makes it a breeze to interface with all of the integrated hardware. The 3pi is also compatible with the popular Arduino development platform.

5.8 Symmetry breaking in social tasks

The approach can be also applied to my model, with a few variations, and I can describe the procedure that could be used in the future to verify the symmetry breaking property. By keeping the same notation N agents are faced with a binary decision between being a S (seeker) or a P (parasite). The symmetry parameter is defined as $s(t) = L(t)/N$, where L is the number of agents which became seekers at time t. A majority decision is any steady state of the system where at least $L \geq \delta N$ agents have became seekers with $0.5 \ll \delta \leq 1.0$. That means that $s \geq \delta$ and s is the only parameter required because $s + p = N$ and thus $p(t)$ is not required. If $s(t)$ converges as demonstrated in the previous experiments, it is always possible to find the steady state probability density function (PDF) for $s(t)$ noted as $p^*(s)$. The PDF $p^*(s)$ estimates how many agents will become seekers. By integrating the PDF function we can estimate the proportion P of experiments in which a majority decision with at least δ majority occurs:

$$\int_0^{\delta} p^*(s)\mathrm{d}s + \int_{1-\delta}^{1} p^*(s)\mathrm{d}s = P \tag{5.31}$$

To find $p^*(s)$ is necessary to run a large number of parallel simulations with different initial conditions and then to run a statistical analysis. To avoid this computational cost, the author Hamann et al. (2010a) makes an assumption about the nature of $s(t)$ by assuming that is described by a mono dimensional Langevin equation, which is a form of stochastic differential equation:

$$\frac{ds}{dt} = \alpha(s,t) + \beta(s,t)\xi(t) \tag{5.32}$$

where:

- α is the deterministic development or drift
- ξ is the Gaussian noise $|\xi(t)| = 1$, with mean $< \xi(t) >= 0$, and uncorrelated in time $< \xi(t)\xi(t') >= \delta(t - t')$
- β is the fluctuation of the the noise amplitude

It is not possible to assume that such a mono dimensional description exists for every high dimensional system, thus the approach assumes that such a description exists and then an estimation of α, β is feasible with some heuristic formula or with some numerical fitting strategies. Once the two parameters are estimated we need to accept or reject the hypothesis by comparing the statistical property of $s(t)$ generated by the detailed simulations with the solution $s(t)$ of the Langevin equation with the estimated parameters.

The most common heuristic to estimate the parameters was used in Hamann et al. (2010b) and is based on two features of $s(t)$. The first is the mean of the absolute changes:

$$\Delta s^{abs}(s,t) = \frac{1}{K}\sum_i |s_i(t) - s_i(t-1)| \tag{5.33}$$

averaged over K samples $s_i(t)$ from many independent simulations runs. The second is the mean of the relative changes:

$$\Delta s^{rel}(s,t) = \frac{1}{K}\sum_i s_i(t) - s_i(t-1) \tag{5.34}$$

which is an approximation of the derivative because it contains the difference between two time steps. Unfortunately the author did not have any time left for running additional simulations and computing the mean and absolute and relative change so he can only speculate about what their outcome would be. The $\Delta s^{rel}(s,t)$ gives an indication of the stability of the system by computing how many zero crossings the function has. Each zero crossing indicates that the derivative is zero and thus a steady state was achieved for that particular configuration. The heuristic to estimate α, β is built on the discretised version of the Langevin equation 5.32:

$$s_{t+1} = s_t + \Delta s^{rel}(s_t) + (\Delta s^{abs}(s_t) - |\Delta s^{rel}(s_t)|)\xi_t \tag{5.35}$$

where ξ_t is again the Gaussian white noise. The white noise is a general approximation but one could calculate the second moments of Δ like variance for each time step. The FokkerPlanck equation can be used for computing the probability density for a stochastic process described by a stochastic differential equation:

$$\frac{\partial \rho_s}{\partial t} = \frac{\partial}{\partial s}(\alpha(s,t)\rho_s) + \frac{1}{2}\frac{\partial^2}{\partial s^2}(\beta^2(s,t)\rho_s) \tag{5.36}$$

to obtain the time development of the probability density function for s and thus its steady state PDF. The parameters (drift and diffusion coefficient) are thus:

$$\alpha(s,t) = \Delta s^{rel}(s_t)\beta(s,t) = \Delta s^{abs}(s_t) - |\Delta s^{rel}(s_t)|$$

To validate the model, one needs to compute the PDF from the simulations and from the solution of the Fokker-Plank equation. If the model is valid we should see a bifurfaction diagram. If the model is not valid, there are several possible explanations:

- the heuristic to compute α, β was not valid
- the system cannot be described by a mono dimensional Langevin equation

The Fokker-Planck equation -if valid- can then be used to quantify the effectiveness of the symmetry breaking by computing the steady state $p^*(s) = 0$ and the stability of the decision by computing the splitting probabilities $\pi_w(x)$.

In summary, the statistic modelling approach is a powerful tool for the analysis of self-organising systems. However it cannot be applied to systems where there are more then two decisions.

Introduction

Every day we make predictions based on our personal subjective experiences. Our predictions about durations or extent of everyday phenomena such as human life spans and the box-office take of movies are based on an optimal Bayesian model (Griffiths and Tenenbaum, 2006). Human perceptions, memory, inferring a 3D structure from a 2D structure, judging when a particular fact is likely to happen in the future are also based on approximate statistical inference.

It is possible then, that we use Bayesian inference when bidding on the final price of an auction. There are 3 main type of auctions known in literature(Shor, 2012):

- First Price Auction: an auction in which the bidder who submitted the highest bid is awarded the object being sold and pays a price equal to the amount bid.
- Second Price Auction: an auction in which the bidder who submitted the highest bid is awarded the object being sold and pays a price equal to the second highest amount bid.
- English Auction: a type of sequential second price auction in which an auctioneer directs participants to beat the current, standing bid. New bids must increase the current bid by a predefined increment. The auction ends when no participant is willing to outbid the current standing bid. Then, the participant who placed the current bid is the winner and pays the amount bid.

First and Second Price Auctions follow a Bayesian Nash Equilibrium, but there are no models of English price auction based on ABM (agent based model) where agents make prediction about the final price and also about other's expectations.

I assume that in English price auctions, players uses Bayesian inference to estimate the final price v_{final} of an auction b. For clarity of notation I will use p to indicate probabilities, v to indicate prices (values), a_i to indicate an agent with $i = 1, ..., n$, b to indicate an auction. The task of the agent $\forall a_i$ is to estimate v_{final} from the current price v of the auction b. The Bayesian predictor computes a probability distribution over v_{final} given v, by applying the Baye's rule:

$$p(v_{final}|v) \propto p(v|v_{final})p(v_{final}) \tag{5.37}$$

The conditioned probability of the event v_{final} given the actual price v is proportional to likelihood $p(v|v_{final})$ and the prior probability $p(v_{final})$. The likelihood is the probability of observing during an auction b a bid of price v given that the final price of the auction is v_{final}. We assume that agents are equally likely to observe any price of an auction in the range $[0, v_{final}]$, which means using a uniform random sampling rate.

$$p(v|v_{final}) = 1/v_{final}, v < v_{final} \tag{5.38}$$

$$p(v|v_{final}) = 0, v > v_{final} \tag{5.39}$$

The same assumption was also used in (Griffiths and Tenenbaum, 2006) for the duration of life spans but also in Bayesian models of visual perception. The prior probability $p(v_{final})$ models the agent's expectation about the final price of the auction b. A statistical analysis of online ebay auctions shows that there are two main classes of final price distributions. A class of products -such as cars- follow a Gaussian distribution with mean μ and standard deviation σ where others -such as handy crafts- follow a gamma distribution with scale parameter θ and shape parameter k. The difference in distribution could be caused by the fact that the price of commodity gods such as cars can be assessed precisely, whereas for other niche gods the price distribution has a long tail due to over estimation. Combining the prior with the likelihood according to Eq.5.39 yields a probability of $p(v_{final}|v)$ over all possible final prices for an auction with a current value of v. A good guess for v_{final} is thus the median of this distribution, called a Bayesian prediction function which estimates the final price of an auction given the current price. Every agent a_i has thus a prediction function:

$$Pred_i : v \to v_{final} \tag{5.40}$$

which allows the estimation of the final price of an auction given the current price. Gaussian prior have non linear prediction functions, whereas Erlang priors have linear prediction functions. Evert agent has also an expectation function:

$$Exp_{i,j} : v \rightarrow v_{final} \tag{5.41}$$

which models the expectation that the agent i has toward the prediction function of the agent j. The expectation function is computed by the agent during the auction and is a point estimate of the others' prediction functions. It also necessary to introduce an energy function $E_i : t \rightarrow p_{bid}$ for every agent i which models the risk associated with the bidding:

$$E_i = \frac{p_{max}}{1 + exp(-m_i \cdot t)} \tag{5.42}$$

where $p_{max} = 1$ for normalization, p_{bid} is the probability of bidding and m_i is the sensitivity to the risk. The logit function is based on the typical behaviour of online auctions where the rate of bidding is increasing toward the end of the auction. The energy function is essentially a reward signal similar to the one used in the Q-learning learning agent.

Virtual auction model

In a virtual auction there are N agents modelled as grey boxes. Every agent $i = 1, ..., N$ has a prediction function $Pred_i$ and an expectation function $Exp_{i,j}$ with $i \neq j$. Agents bid in sequence at regular intervals, the auction starts with an initial price of 0 and stops after T time steps. At each time step the agent can decide whether to bid or not. Every agent has equal buying power but could have different prediction functions according to their subjective knowledge (a priori distribution). Each agent's goal is to win the bid, which means to predict or estimate the final price of the bid according to the actual price. The disturbance is generated by other agents bidding in temporal sequence. Agents are not allowed to communicate with external channels or to bid on multiple auctions. Every auction is independent from each other and agents learn from each auction experience. As an explicative example I show that by choosing an Erlang prediction function (see Eq.5.57) for $N = 2$ agents without expectations, the time series has a linear behaviour. In the second example if I choose a Gaussian prediction function (see Fig.5.33) for $N = 2$ agents without expectations, the time series has a non linear behaviour.

Discussion

The model could be validated in accuracy over 2 parameters:

- final price prediction: prediction of the final price of an item when N human agents are bidding
- time series similarity: trajectories of price evolution during bidding

Firstly one could measure the ability of the model to predict the final price of an an auction by only providing the parameter N and the coefficient of dissimilarity between players C. In particular given a new ebay auction where we only know the initial price and the number of bidders the model would predict the final price with a confidence interval of ξ.

Secondly one could measure the similarity of the trajectories produced by the model with real online auctions. For this purpose, it is possible to collect auction data from Ebay, one of the biggest online auction websites. The data should be filtered to remove multiple bidders (the same player bids on different auctions) and automated “snipers“ (softwares agent that bid at the last moment before an auction closes). It should not be necessary to filter the automatic bidding function: a player can set a limit budget and the ebay system will incrementally bid until the limit is reached. These are special cases of bidding behaviour which are not reproduced in the model:

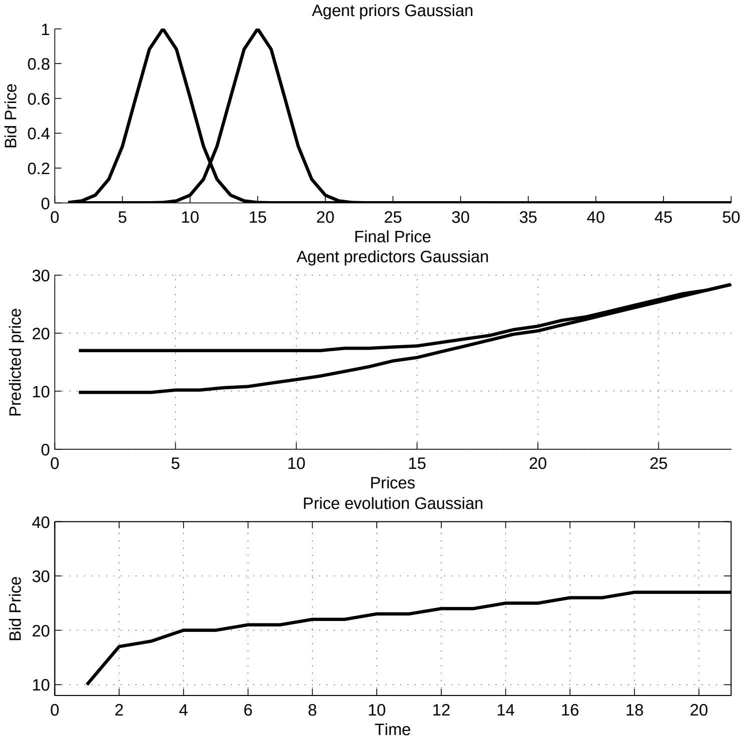


Figure 5.30: Two virtual agents using a Gaussian prediction function

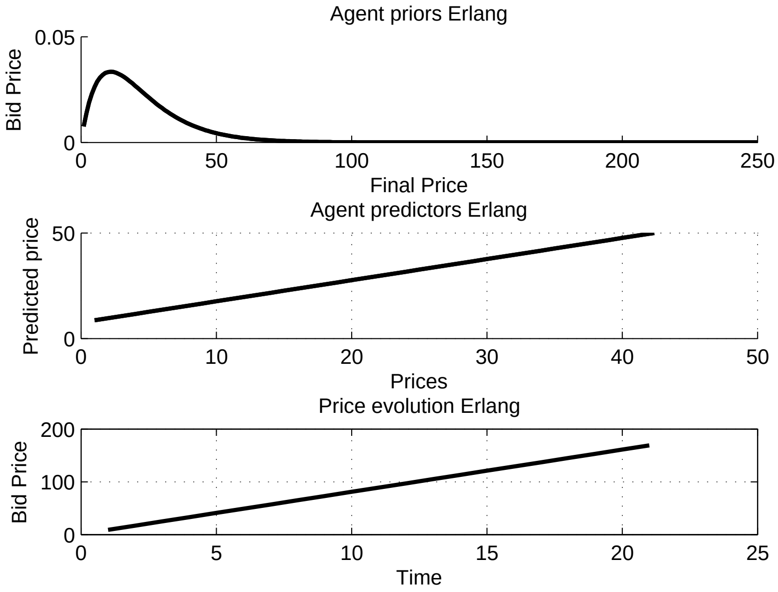


Figure 5.31: Two virtual agents using an Erlang prediction function.

- multi bid: agents bets on parallel auctions thus maximising their chances of winning the same item at lower prices
- uniform sampling of prices
- offline learning: during the auction, the agents uses the static predictive function and expectations, they only update their a priori distribution between bidding sessions

A priory biasing is an important factor for symmetry breaking, when all the players have the same subjective knowledge the time series are linear, in the case of Erlang distributions and to some extent in the Gaussian distributions. But expectations will play an important role in generating the necessary dynamic during the bidding process. In summary, this simplified learning model could predict the outcome of ebay auctions and therefore can be extended to similar economic games where prediction is essential like the stock market.

5.9 Bayesian inference

Baye's rule for a uniform random distributed price $[0, v_{final}]$

$$p(v_{final}|v) = \frac{p(v|v_{final})p(v_{final})}{p(v)} \tag{5.43}$$

where:

$$p(v) = \int_0^\infty p(v|v_{final})p(v_{final})dv_{final} \tag{5.44}$$

Because v is sampled uniformly at random (See Eq.3.2), the previous equation becomes:

$$p(v) = \int_v^\infty \frac{p(v_{final})}{v_{final}} dv_{final} \tag{5.45}$$

and is only dependent on the prior distribution.

Once we have computed $p(v_{final}|v)$, the prediction function can be generated by finding the posterior median v^*:

$$P(v_{final} > v^*|v) = 0.5 \tag{5.46}$$

$$P(v_{final} > v^*|v) = \int_{v^*}^\infty p(v_{final}|v)dv_{final} \tag{5.47}$$

The point v^* can be computed analytically for the gamma or Erlang distribution and numerically for the Gaussian distribution.

5.9.1 Gamma prediction function

The Gamma prior distribution follows:

$$p(v_{final}) \propto v_{final}^{k-1} e^{-v_{final}/\beta} \tag{5.48}$$

The posterior distribution is:

$$p(v) \propto \int_v^\infty v_{final}^{k-2} e^{-v_{final}/\beta} dv_{final} \tag{5.49}$$

$$= -v^{k-1} \cdot E_{2-k}(\frac{v}{\beta}) \tag{5.50}$$

Where the E_{2-k} is the exponential integral function as in Fig.5.32.

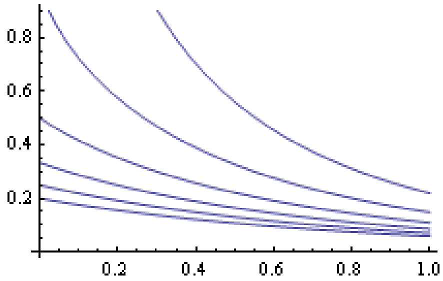


Figure 5.32: Integration values for the Gamma function

5.9.2 Erlang prediction function

The Erlang prior distribution is a Gamma prior with k a chosen integer. For example a $k = 2$ gives:

$$p(v_{final}) \propto v_{final} e^{-v_{final}/\beta} \tag{5.51}$$

The posterior distribution is:

$$\begin{aligned} p(v) &\propto \int_v^\infty e^{-v_{final}/\beta} dv_{final} && (5.52) \\ &= -\beta e^{-v_{total}/\beta} |_v^\infty && (5.53) \\ &= \beta e^{-v_/\beta} && (5.54) \end{aligned}$$

The posterior median is:

$$\begin{aligned} P(v_{final} > v^*|v) &= \int_{v^*}^\infty \frac{1}{\beta} e^{-(v_{final}-v)/\beta} dv_{final} && (5.55) \\ &= e^{-(v^*-v)/\beta} && (5.56) \end{aligned}$$

And we can find v^* by imposing $e^{-(v^*-v)/\beta} = 0.5$, thus obtaining a linear prediction function with unitary slope and intercept $\beta log 2$:

$$v^* = v + \beta log 2 \tag{5.57}$$

5.9.3 Gaussian prediction function

The Gaussian prior distribution:

$$p(v_{final}) \propto e^{-\frac{(v_{final}-\mu)^2}{2\sigma^2}} \tag{5.58}$$

The posterior distribution of the Gaussian prior has no simple analytical form:

$$p(v) \propto \int_v^\infty \frac{1}{v_{final}} e^{-\frac{(v_{final}-\mu)^2}{2\sigma^2}} dv_{final} \tag{5.59}$$

Therefore to compute $p(v)$ we need to use a numerical integration method, Fig.5.33 shows the numerical integration results:

The prediction function is also computed with numerical integration, plus optimisation. Some prediction functions are shown in Fig.5.34.

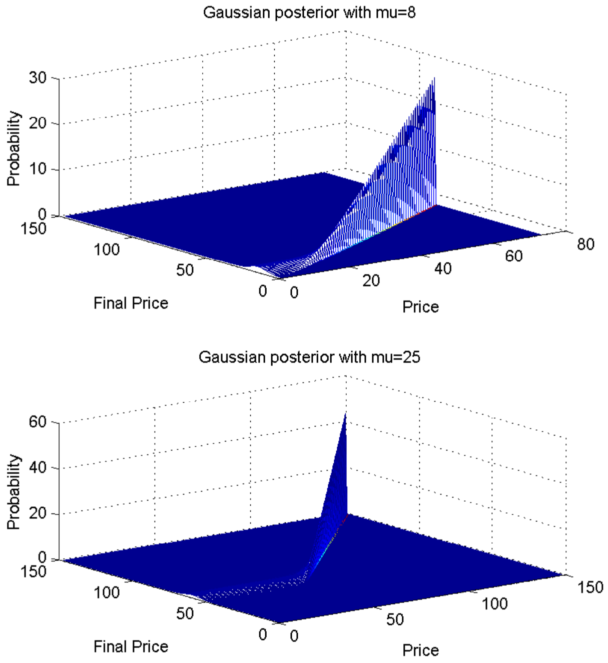


Figure 5.33: Gaussian posterior distribution computed via numerical integration.

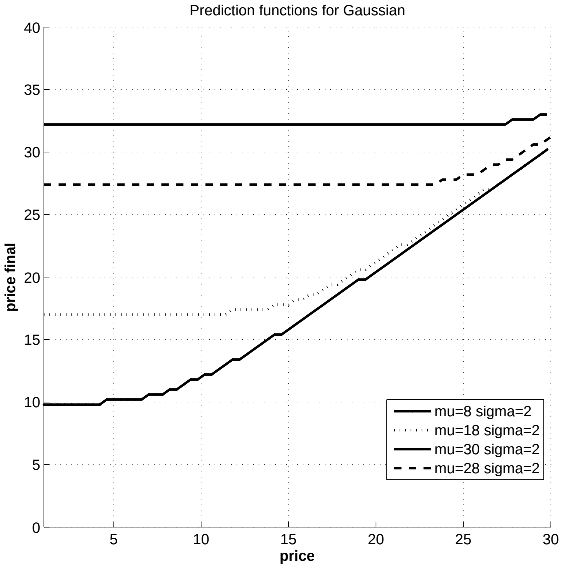


Figure 5.34: Gaussian predictions functions computed via numerical integration.

Bibliography

Arimoto, S. (1972). An algorithm for computing the capacity of arbitrary memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20.

Ashby, W. R. (1956). *An introduction to Cybernetics*. Chapmann and Hall Ltd.

Axelrod, R. (1984). *The Evolution of Cooperation*. Basic Books.

Ay, N., Bertschinger, N., Der, R., Güttler, F., and Olbrich, E. (2008). Predictive information and explorative behavior of autonomous robots. *The European Physical Journal B - Condensed Matter and Complex Systems*, 63:329–339. 10.1140/epjb/e2008-00175-0.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1990). *Neuronlike adaptive elements that can solve difficult learning control problems*, pages 81–93. IEEE Press, Piscataway, NJ, USA.

Beekman, M., Nicolis, S., Meyer, B., and Dussutour, A. (2009). Noise improves collective decision-making by ants in dynamic environments. In *Proceedings in Biological Science*, volume 22.

Beni, G. and J.Wang (1989). Swarm intelligence in cellular robotic systems. *In NATO Advanced Workshop on Robotics and Biological Systems*, pages 26–30.

Berger, C. R. and Calabrese, R. J. (1975). Some explorations in initial interaction and beyond: Toward a developmental theory of interpersonal communication. *Human Communication Research*, 1(2):99–112.

Bertsekas, D. P. and Tsitsiklis, J. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bialek, W., Nemenman, I., and Tishby, N. (2001). Predictability, complexity, and learning. *Neural Computation*, 13(11):2409–2463.

Blahut, R. (1972). Computation of channel capacity and rate distorsion functions. *IEEE Transactions on Information Theory*, 18(4):460–473.

Blais, B. S., Intrator, N., Shouval, H., and Cooper, L. N. (1998). Receptive field formation in natural scene environments: Comparison of single cell learning rules. *Neural Computation*, 10:1797–1813.

Bonomi, A., Manzoni, S., Pisano, A., and Vizzari, G. (2009). Experimenting situated cellular agents in indoor scenario: Pedestrian dynamics during lecture hall evacuation. In *Web Intelligence/IAT Workshops*, pages 591–594. IEEE.

Booth, T. L. (1967). *Sequential Machines and Automata Theory*. John Wiley and Sons Inc., 1 edition.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*. Bradford, Colorado.

Brembs, B. (1996). Chaos, cheating and cooperation: potential solutions to the prisoner's dilemma. *Oikos: A Journal of Ecology*, 76(1):14–24.

Buccino, G., Vogt, S., Ritzl, A., Fink, G. R., Zilles, K., Freund, H.-J., and Rizzolatti, G. (2004). Neural circuits underlying imitation learning of hand actions: An event-related fMRI study. *Neuron*, 42(2):323–334.

Byrne, R. W. and Whiten, A. (1988). *Machiavellian intelligence: Social expertise and the evolution of intellect in monkeys, apes, and humans.* Oxford Science Publications.

Camazine, S., Franks, N. R., Sneyd, J., Bonabeau, E., Deneubourg, J.-L., and Theraula, G. (2001). *Self-Organization in Biological Systems.* Princeton University Press, Princeton, NJ, USA.

Dennett, D. C. (1987). *The Intentional Stance.* MIT Press.

Der, R., Güttler, F., and Ay, N. (2008). Predictive information and emergent cooperativity in a chain of mobile robots. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 166–172. MIT Press, Cambridge, MA.

Di Prodi, P., Porr, B., and Wörgötter, F. (2008). Adaptive communication promotes sub-system formation in a multi agent system with limited resources. *LAB-RS '08: Proceedings of the 2008 ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*, pages 89–96.

Dittrich, P., Kron, T., and Banzhaf, W. (2003). On the scalability of social order - modeling the problem of double and multi contingency following Luhmann. *Journal of Artificial Societies and Social Simulation*, 6.

Dorigo, M. and Birattari, M. (2007). Swarm intelligence. *Scholarpedia*, 2(9):1462. URL: `www.scholarpedia.org/article/Swarm_intelligence`, Last Checked: 7 Jan 2012.

Durkheim, É. (1893). *The Division of Labor in Society.* The Free Press, 1230 Avenue of the Americas, New York.

Ernst, I. (1925). Beitrag zur theorie des ferromagnetismus. *Physik*, 31:253–258.

Floreano, D., Mitri, S., and Magnenat, S. (2007). Evolutionary conditions for the emergence of communication in robots. *Current biology*, 17(17):514–519.

Foerster, H. (1960). On self-organizing systems and their environments. In Yovits, M. and Cameron, S., editors, *Self-Organizing Systems*, pages 31–50. Pergamon Press, London.

Franks, N. R., Mallon, E. B., Bray, H. E., Hamilton, M. J., and Mischler, T. C. (2003). Strategies for choosing between alternatives with different attributes: exemplified by house-hunting ants. *Animal Behaviour*, 65(1):215–223.

Gardner, M. (1970). Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223:120–123.

Georgoudas, I. G., Sirakoulis, G. C., and Andreadis, I. (2006). A simulation tool for modelling pedestrian dynamics during evacuation of large areas. In Maglogiannis, I., Karpouzis, K., and Bramer, M., editors, *AIAI*, volume 204 of *IFIP*, pages 618–626. Springer.

Gibson, J. J. and J., E. (1955). Perceptual learning: differentiation or enrichment? *Psychological Review*, 62:32–41.

Gilbert, N. and Conte, R. (1995). *Artificial Societies: The Computer Simulation of Social Life*. Routledge.

Griffiths, T. and Tenenbaum, J. (2006). Optimal predictions in everyday cognition. *Psychological Science*, 17(9):767–727.

Guttman, R. H., Moukas, A. G., and Maes, P. (1999). Agents as mediators in electronic commerce. *In Intelligent Information Agents*, pages 131–152.

Haitao, O., Weidong, Z., Wenyuan, Z., and Xiaoming, X. (2000). A novel multi-agent Q-learning algorithm in cooperative multi-agentsystem. *Intelligent Control and Automation. Proceedings of the 3rd World Congress*, 1:272 – 276.

Hamann, H., Meyer, B., Schmickl, T., and Crailsheim, K. (2010a). A model of symmetry breaking in collective decision-making. In *Proceedings of the 11th international conference on Simulation of adaptive behavior: from animals to animats*, SAB2010, pages 639–648, Berlin, Heidelberg. Springer-Verlag.

Hamann, H., Schmickl, T., Wrn, H., and Crailsheim, K. (2010b). Analysis of emergent symmetry breaking in collective decision making. *Neural Computing & Applications*, pages 1–12. 10.1007/s00521-010-0368-6.

Hiroaki Kitano, S. H. (1997). The virtual biology laboratories: A new approach of computational biology. *Proceedings of the Fourth European Conference on Artificial Life*, pages 274–283.

Hü, M. and Pasemann, F. (2002). Dynamical Neural Schmitt Trigger for Robot Control. *Lecture Notes in Computer Science*, 2415:142–142.

Hülse, W. P. (2004). Structure and function of evolved neuro-controllers for autonomous robots. *Connections Science*, 16(4):249–266.

Hurri, J. and Hyvärinen, A. (2003). Simple-cell-like receptive fields maximize temporal coherence in natural video.

Kennedy, J. N. and Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. *Proceedings of the 2001 Congress on Evolutionary Computation*, 1:81–86.

Kernbach, S., Thenius, R., Kornienko, O., and Schmickl, T. (2009). Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic swarm. *Adaptive Behavior*, 17:237–259.

Kirchner, A. and Schadschneider, A. (2002). Cellular automaton simulations of pedestrian dynamics and evacuation processes. URL: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.8.9008`, Last Visited 7 Jan 2012.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2004). Organization of the information flow in the perception-action loop of evolved agents. In *2004 NASA/DoD Conference on Evolvable Hardware. IEEE Computer Society*, pages 177–180.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). Empowerment: A universal agent-centric measure of control. In *IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 128–135.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2007). Representations of Space and Time in the Maximization of Information Flow in the Perception-Action Loop. *Neural Comp.*, 19(9):2387–2432.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2008). Keep your options open: An information-based driving principle for sensorimotor systems. *PLoS ONE*, 3(12):e4018.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems of Information and Transmission*, 1(1):1–7.

Kolodziejski, C. and Kulvicius, T. (2009). On the analysis of differential hebbian learning in closed-loop behavioral systems. *Frontiers in Computational Neuroscience. Conference Abstract: Bernstein Conference on Computational Neuroscience.*

Kulvicius, T., Kolodziejski, C., Tamosiunaite, M., Porr, B., and Wörgötter, F. (2010). Behavioral analysis of differential hebbian learning in closed-loop systems. *Biological Cybernetics*, 103:255–271.

Kulvicius, T., Porr, B., and Wörgötter, F. (2007). Development of receptive fields in a closed-loop behavioural system. *Neurocomputing*, 70(10-12):2046–2049. Computational Neuroscience: Trends in Research 2007, Computational Neuroscience 2006.

Krding, K. P., Knig, P., P, K., Kayser, C., Kayser, C., Einhuser, W., and Einhuser, W. (2004). How are complex cell properties adapted to the statistics of natural stimuli? *Journal of Neurophysiology*, 91:2004.

Lai, C. S. L., Fisher, S. E., and Hurst, J. A. (2001). A forkhead-domain gene is mutated in a severe speech and language disorder. *Nature*, 6855:519–23.

Lin, L.-J. and Mitchell, T. M. (1992). Memory approaches to reinforcement learning in non-markovian domains. *Artificial Intelligence*, 8(CMU-CS-92-138):293–321.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20:46–61.

Luhmann, N. (1984). *Soziale Systeme*. Suhrkamp, Frankfurt am Main.

Luhmann, N. (1993). Soziologische aufklärung 3: Soziales system, gesellschaft, organisation. *Opladen: Westdeutscher Verlag.*

Luhmann, N. (1995). *Social Systems*. Stanford University Press, Stanford, California.

Luhmann, N. (2000). Organisation und entscheidung. *Opladen: Westdeutscher Verlag.*

Lungarella, M., Pegors, T., Bulwinkle, D., and Sporns, O. (2005). Methods for quantifying the informational structure of sensory and motor data. *Neuroinformatics*, 3:243–262. 10.1385/NI:3:3:243.

Lungarella, M. and Sporns, O. (2006). Mapping information flow in sensorimotor networks. *PLoS Computational Biology*, 2(10):e144.

MacDermot, K., Bonora, E., and Sykes, N. (2005). Identification of FOXP2 truncation as a novel cause of developmental speech and language deficits. *The American Journal of Human Genetics*, 6:1074–80.

Marrocco, D. and Nolfi, S. (2006). Origins of communication in evolving robots. *From Animals to Animats*, 1(9):789–803.

McFarland, D. J. (1993). *Intelligent behavior in animals and robots.* MIT Press, Cambridge, MA.

McKinstry, J. L., Edelman, G. M., and Krichmar, J. L. (2006). A cerebellar model for predictive motor control tested in a brain-based device. *Proceedings of National Academic Science U S A*, 103(9):3387–3392.

Meyer, B., Beekman, M., and Dussutour, A. (2008). Noise-induced adaptive decision-making in ant-foraging. In *SAB'08*, pages 415–425.

Moller, A. (1988). False alarm calls as a means of resource usurpation in the Great Tit Parus-Major. *ETHOLOGY*, 79(1):25–30.

Niedermeyer, E. and da Silva, L. F. (2004). *Electroencephalography: Basic Principles, Clinical Applications and Related Fields*. Lippincott Williams & Wilkins, fifth edition.

Nolfi, S. (2004/2005). Behaviour as a complex adaptive system: On the role of self-organization in the development of individual and collective behaviour. *Complexus*, 2:195–203.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609.

Parsons, T. (1937). *The Structure of Social Action*. McGraw-Hill, New York, second edition.

Parsons, T. (1951). *The Social System*. Routledge & Kegan Paul Ltd, London and Henley.

Parsons, T. (1977). *Social systems and the evolution of action theory*. The Free Press, New York.

Pegors, T., Bulwinkle, D., and Lungarella, M. (2005). Methods for quantifying the information structure of sensory and motor data. *Neuroinformatics*, 3:343–262.

Pfeifer, R., Lungarella, M., Sporns, O., and Kuniyoshi, Y. (2008). On the information theoretic implications of embodiment - principles and methods. *Proc. of the 50th Anniversary Summit of Artificial Intelligence*, pages 76–86.

Polani, D. (2010). Information flows in causal networks. *Social Science Research Network Working Paper Series*.

Polani, D., Kim, J. T., and Martinetz, T. (2001). An information-theoretic approach for the quantification of relevance. In Kelemen, J. and Sosik, P., editors, *Advances in Artificial Life*. Springer.

Polani, D., Klyubin, A. S., and Nehaniv, C. L. (2004). Organization of the information flow in the perception-action loop of evolved agents. *Evolvable Hardware, Proceedings.*, pages 177–180.

Polani, D., Nehaniv, C. L., and Klyubin, A. S. (2005). Empowerment: A universal agent-centric measure of control. *Proceedings of the IEEE Congress on Evolutionary Computation*, 1:128–135.

Porr, B., Egerton, A., and Wörgötter, F. (2006). Towards closed loop information: Predictive information. *Constructivist Foundations*, 1(2):83–90.

Porr, B. and Wörgötter, F. (2003). Isotropic sequence order learning. *Neural Computation*, 15:831–864.

Porr, B. and Wörgötter, F. (2003). Isotropic sequence order learning in a closed loop behavioural system. *Roy. Soc. Phil. Trans. Math., Phys. & Eng. Sciences*, 361(1811):2225–2244.

Porr, B. and Wörgötter, F. (2005). Inside Embodiment - What means Embodiment to Radical Constructivists? *Kybernetes*, pages 105–117.

Porr, B. and Wörgötter, F. (2006). Strongly improved stability and faster convergence of temporal sequence learning by utilising input correlations only. *Neural Computation*, 18(6):1380–1412.

Porr, B. and Wörgötter, F. (2007). Fast heterosynaptic learning in a robot food retrieval task inspired by the limbic system. *Biosystems*, 89:294–299.

Prokopenko, M., Gerasimov, V., and Tanev, I. (2006). Evolving spatiotemporal coordination in a modular robotic system. In *SAB 2006*, pages 558–569.

Quiroga, R. Q., Kraskov, A., Kreuz, T., and Grassberger, P. (2002). Performance of different synchronization measures in real data: A case study on electroencephalographic signals. *Physical Review E*, 65(4):041903.

Reading, N. C. and Sperandio, V. (2006). Quorum sensing: the many languages of bacteria. *FEMS Microbiology Letters*, 254(1):1–11.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry.* New Jersey: World Scientific Publishing Company.

Roulston, M. (1999). Estimating the errors on measured entropy and mutual information. *Physica D*, 125:285–294.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation. *IEEE Transactions on Autonomous Mental Development,*, 2(3):230–247.

Schreiber, T. (2000). Measuring information transfer. *Phys Rev Lett*, 85:461–464.

Schwieren, C. and Weichselbaumer, D. (2010). Does competition enhance performance or cheating? a laboratory experiment. *Journal of Economic Psychology*, 31(3):241 – 253.

Seyfarth, R. M. and Cheney, D. L. (2000). Social awareness in monkeys. *American Zoologist*, 40(6):902–909.

Shannon, C. E. and Weaver, W. (1949). *The mathematical theory of communication.* University of Illinois Press, Urbana.

Shor, M. (2012). Dictionary of game theory terms. `http://www.gametheory.net/`. Last visited 9 Jan 2012.

Smith, A. (1776). *The Wealth of Nations.* Bantam Classics (March 4,2003).

Smith, J. M. and Harper, D. (2003). *Animal Signals.* Oxfold University Press.

Stamm, K. (2006). Individual learning and the dynamics in predator-prey populations. *Göttingen informatic journal*, (ZFI-NM-2007-08):243–259.

Steels, L. (1998). The origins of ontologies and communication conventions in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 2(1):169–194.

Steels, L. (1999). *The Talking Heads Experiment.* Laboratorium, Antwerpen, Belgium. Limited Preedition.

Stone, P. and Veloso, M. (1998). A Layered Approach to Learning Client Behaviors in the RoboCup Soccer Server. *Applied Artificial Intelligence*, 12:165–188.

Stuart, B. (1984). Nicholas Minorsky and the automatic steering of ships. *Control Systems Magazine, IEEE*, 4(4).

Sugawara, T. and Lesser, V. (1998). Learning to Improve Coordinated Actions in Cooperative Distributed Problem-Solving Environments. *Machine Learning*, 33:129–153.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* A Bradford Book.

Talcott, P. (1967). *Sociological Theory and Modern Society.* Free Press, first edition.

Tan, M. (1997). *Multi-Agent Reinforcement Learning: Independent vs. Cooperative Learning.* Morgan Kaufmann, San Francisco, CA, USA.

Thomas, H. (1885). *Leviathan, or, The matter, forme, and power of a common-wealth ecclesiasticall and civil.* G. Routledge and sons.

Tishby, N., Pereira, F., and Bialek, W. (1999). The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

Tomas, K., Christoph, K., Minija, T., Bernd, P., and Florentin, W. (2010). Behavioral analysis of differential hebbian learning in closed-loop systems. *Biological Cybernetics*, 103:255–271. 10.1007/s00422-010-0396-4.

Touchette, H. and Lloyd, S. (2000a). Information-theoretic approach to the study of control systems. *Physica A*, 331:140–172.

Touchette, H. and Lloyd, S. (2000b). Information-theoretic limits of control. *Physical Review Letters*, 84(6):1156–1159.

Touchette, H. and Lloyd, S. (2004). Information-theoretic approach to the study of control systems. *Physica A*, 331:140–172.

Vapnik, V. N. (1998). *Statistical Learning Theory.* Wiley-Interscience.

Varas, A., Cornejo, M., Mainemer, D., Toledo, B., Rogan, J., Munoz, V., and Valdivia, J. (2007). Cellular automaton model for evacuation process with obstacles. *Physica A: Statistical Mechanics and its Applications*, 382(2):631–642.

Varela, F. J. and Maturana, H. R. (1980). *Autopoiesis and Cognition: The Realization of the Living*, volume 42. D. Reidel Publishing Company.

Čapek, K. and Novack, C. (1920). *R.U.R. (Rossum's universal robots).* Penguin Classics.

Verschure, P. and Coolen, A. (1991). Adaptive fields: Distributed representations of classically conditioned associations. *Network*, 2:189–206.

Verschure, P. and Voegtlin, T. (1998). A bottom-up approach towards the acquisition, retention, and expression of sequential representations: Distributed adaptive control III. *Neural Networks*, 11:1531–1549.

Verschure, P. F. (1998). Synthetic epistemology: The aquisition, retention, and expression of knowledge in natural and synthetic systems. In *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*, pages 147–153, Anchorage. IEEE.

Verschure, P. F. M. J., Voegtlin, T., and Douglas, R. J. (2003). Environmentally mediated synergy between perception and behaviour in mobile robots. *Nature*, 425:620–624.

Volterra, V. (1931). Variations and fluctuations of the number of individuals in animal species living together. *Animal Ecology*, pages 48–409.

Von Foerster, H. (1985). *Sicht und Einsicht: Versuche zu einer operativen Erkenntnistheorie.* Vieweg, Braunschweig.

Von Foerster, H. (2003). *Understanding understanding: essays on cybernetics and cognition*. Springer-Verlag.

von Glasersfeld, E. (1995). *Radical Constructivism. A way of knowing and learning*. Falmer Press, London.

Wang, G., Zhang, J., and Bi, D. (2005). Novel learning feed-forward controller for accurate robot trajectory tracking. *Lecture Notes in Computer Science*, 3611/2005:266–269.

Watkins, C. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4):279–292.

Weber, C. and Obermayer, K. (1999). Orientation Selective Cells Emerge in a Sparsely Coding Boltzmann Machine. In *Proceedings ICANN*, pages 286–291.

Weisbuch, G. and Stauffer, D. (2000). Hits and flops dynamics. Working Papers 00-07-036, Santa Fe Institute.

Werner, J. and Buse, M. (1989). Closed loop control of human body temperature: results from a one-dimensional model. *Biological Cybernetics*, 61:467–75.

Wiener, N. (1961). *Cybernetics — or control and communication in the animal and the machine*. The M.I.T. Press, Cambridge, Massachusetts, 2 edition.

Wimmer, H. and Perner, J. (1983). Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception. *Cognition*, 13:103–128.

Wischmann, S. and Pasemann, F. (2006). The emergence of communication by evolving dynamical systems. *From Animals to Animats*, 1(9):777–788.

W.Macy, M. and Willer, R. (2002). From factors to actors: Computational sociology and agent-based modeling. *Annual Review of Sociology*, 28:143–166.

Wolfram, S. (1994). *Cellular Automata and Complexity: Collected Papers*. Stephen Wolfram, LLC.

Wooldrige., M. (2002). An introduction to multiagent systems. *Proceed. NATO Advanced Workshop on Robots and Biological Systems*, pages 243–266.

Wyss, R., König, P., and Verschure, P. (2006). A model of the ventral visual system based on temporal stability and local memory. *PLoS Biology*, 4(5):e120.

Zahavi, A. (1975). Mate selection - a selection for a handicap. *Journal of Theoretical Biology*, 53:205–214.

Zong, X., Xiong, S., Fang, Z., and Li, Q. (2010). Multi-objective optimization for massive pedestrian evacuation using ant colony algorithm. In Tan, Y., Shi, Y., and Tan, K. C., editors, *ICSI (1)*, volume 6145 of *Lecture Notes in Computer Science*, pages 636–642. Springer.