School of Computing
Blekinge Institute of Technology

# Generic Cognitive Architecture for Real-Time, Embedded Cognitive Systems

**Jekaterina Novikova**

**Thesis submitted for completion of Master of Science (60 credits)**
**Main field of study: Computer Science**
**Specialization: Informatics**

**August 2011**

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

**Contact Information:**
Author:
Jekaterina Novikova
E-mail: j.novikova@inbox.lt

University advisor:
Prof. Craig Lindley
School of Computing

School of Computing
Blekinge Institute of Technology
SE-371 41 Karlskrona
Sweden

| | |
|---|---|
| Internet | : www.bth.se/com |
| Phone | : +46 455 38 50 00 |
| Fax | : + 46 455 38 50 57 |

# ABSTRACT

**Context**. The problem of integrated cognition belongs to a multi-disciplinary area of cognitive engineering. The multi-disciplinary focusing on cognitive models and real-time embedded systems, such as mobile robots, helps to reveal a broader and deeper understanding of robotics as part of everyday life and society. Over the past decades many cognitive architectures have been proposed and steadily developed, based on different approaches and methodologies, but still current cognitive architectures are far from the goal of covering the requirements for general intelligence.

Recent research in the area of evolutionary algorithms and genetic programming is used in this study as an inspiration for developing the new version of integrated cognitive architecture, and the knowledge of human brain structure and functions is applied to the architecture as well.

**Objectives**. In this study a survey of cognitive architectures is performed, a version of biologically inspired hybrid cognitive architecture is developed. This architecture is influenced by a contemporary research in evolutionary algorithms and genetic programming. Some modules of the architecture are applied to a mobile robot in a simulated environment.

**Methods**. Several different research methods are used in the thesis, such as literature study, case study and simulated experimentations. The source of the literature, used for the literature study, is Internet and the e-library's resources, freely available for the students. The selection criteria are rely on a trusted-review strategy, that means a review of a subject area published by trusted sources, such as highly ranked journals. The relevance is evaluated by fitness with a purpose of the study. Case study is conducted by investigating the characteristics and abilities of a simple mobile robot in an experimental environment. Experiments of robot's navigating in different environments with some modules of integrated cognitive architecture implemented are conducted in a simulated environment. Robot's wall following behaviour is analysed applying two different genetic programming methods and three different fitness functions. Robot's learning abilities are analysed using different training environments and comparing the navigation output afterwards.

**Results**. The statistical information is gathered during the experiments in order to analyze the performance of a robot's behaviour. For each generation of an evolutionary run average and best population fitness values, complexity of the chromosomes, the number of robot's collisions with other objects and the value of population entropy are stored.

**Conclusions**. The results indicate that the proposed integrated cognitive architecture is plausible and can be implemented successfully to a mobile robot. The implementation of the two modules is described in this thesis, while developing and implementation of other modules is indicated as a future work of the author.

For robot control, the results indicate that genetic programming methods can be successfully integrated with a subsumption architecture and used as a learning mechanism. Different fitness functions, from the other side, can be used for designing relevant navigation behaviour, such as wall following and others. Both genetic programming methods and fitness functions are influencing the quality of robot navigation behaviour. The interaction of these two factors also has an influence on the navigation performance. The results of learning experiments highlight the influence of specific training environments on the future navigational performance.

**Keywords:** integrated cognitive architecture, mobile robot, genetic programming.

# CONTENTS

# LIST OF FIGURES

# LIST OF ACRONYMS

| | |
|---|---|
| ACS | Artificial Cognitive System |
| AI | Artificial Intelligence |
| BICA | Biologically-Inspired Cognitive Architectures |
| CA | Cognitive Architecture |
| GP | Genetic Programming |
| ICA | Integrated Cognitive Architecture |
| MA | Module Acquisition |
| PDP | Parallel Distributed Processing |

## STRUCTURE OF THE THESIS

The thesis is structured into a number of chapters sequentially.

*Chapter 1* is about the background of the topic which is going to be presented in the thesis.

*Chapter 2* is about the problem definition, aims and goals of this research work.

*Chapter 3* discusses the methodology which is adopted during the research work.

*Chapter 4* presents the literature review and theoretical work.

*Chapter 5* is about empirical work, which was done during the process of experiments for the research.

*Chapter 6* presents results, based on the experiments, and the analysis of the results.

*Chapter 7* is the concluding chapter of the thesis with summarized conclusion remarks and suggestions for the future work.

# 1 FIRST CHAPTER: BACKGROUND

The problem, analyzed in the thesis, belongs to the cognitive engineering area, the area which brings together cognitive science (including artificial intelligence), engineering methodologies and practices, information science and design, information retrieval and similar approaches (Roth, Patterson, & Mumaw, 2001). The multi-disciplinary focusing on cognitive models and real-time embedded systems, such as mobile robots, helps to reveal a broader and deeper understanding of robotics as part of everyday life and society.

Within the last years the development of mobile robotics and autonomous intelligent systems has experienced a rapid growth and integrated significantly into the day-to-day lives of humans. Autonomous systems are gradually becoming a part of our way of life, ranging from factory transport systems, airport transport systems, road/vehicular systems, to military applications, automated patrol systems, homeland security surveillance, rescue operations and ambient assistants for elderly or disabled people.

A challenge in the area of cognitive systems and robotics, as stated in the 7th frame program of the EU, is formulated as follows: "Engineering systems with the capability to sense and understand an unstructured environment is a challenge which goes beyond today's systems engineering paradigm. Present day systems engineering relies on specifying every eventuality a system will have to cope with in the execution of its task(s), and programming the appropriate response in each case. The challenge is to extend systems engineering to the design of systems that can carry out useful tasks in circumstances that were not planned for explicitly at design time".

Cognition is the scientific term for "the process of thought", which refers to a faculty for the processing of information, applying knowledge, and changing preferences (Mondada, F., Franzi, E., 1993). Some authors explain a cognitive system from this perspective as a system, which exhibits effective behaviour through perception, action, deliberation, communication, and through either individual or social interaction with the environment (Vernon, 2007). Other authors (e.g. Brachman, 2002) define a cognitive computer system as one which — in addition to being able to reason, to learn from experience, to improve its performance with time, and to respond intelligently to things it's never encountered before — would also be able to explain what it is doing and why it is doing it. The definition of cognition can be expended even more by adding a sense of self-reflection in addition to the characteristics of adaptation and anticipation, as suggested by Hollnagel (1999). Cognition, or cognitive processes, can be natural or artificial, conscious or unconscious.

An integrated cognitive architecture can be defined as a single system that is capable of producing all aspects of behaviour, while remaining constant across various domains and knowledge bases, thus modelling human performance in multimodal multiple task situations (Newell 1990; Anderson et al. 2004). It's important to notice that this definition of cognition covers not only the brain-mediated functions, but also not-mediated ones, such as peripheral nervous system functions. Another definition of integrated cognitive architecture could define it as a system, integrating across a variety of functional modules (Wayne, 2007). Cognitive architecture research seeks to define a collection of integrated processes and knowledge representations that provide a general foundation for cognitive behaviour across an increasingly broad spectrum of problems (B. Goertzel, et al., 2010).

Architectures, in general, have divergent features that lead to different properties, they support specific capabilities and are used in different domains and environments. All these differences are responsible for the variety of architectures. Some authors (Joscha Bach, 2009) divide architectures into the following methodological groups:

- Classical (symbolic) architectures, which are essentially rule-based;
- Parallel distributed processing (PDP) (subsymbolic) architectures;
- Hybrid architectures, which use different layers for different tasks;
- Biologically inspired architectures;
- Emotional and motivational architectures.

Others (Duch W. et al, 2008) group cognitive architectures according to the features of memory organization and knowledge representation schemes into three main groups:

- Symbolic;
- Emergent and
- Hybrid models.

Over the past decades, many cognitive architectures have been proposed and steadily developed, based on different approaches and methodologies, such as Soar (Laird et al. 1986a, b, 1987; Lehman et al. 2006), ACT-R (Anderson et al., 2004), ICARUS (Langley and Choi 2006), BDI (Bratman et al. 1988; Rao and Georgeff 1991), CLARION (Sun and Peterson 1996, 1998; Sun et al. 2001; Sun and Zhang 2006) and others.

Although the impressive progress in many specific sub-topics in AI and Cognitive Science can be recognized, all the current cognitive architectures are far from the goal of covering the requirements for general intelligence. Most systems able to perform complex tasks that humans and other animals can perform easily, have to be very carefully crafted, normally their field of expertise is very narrow, and they are hard to extend. The intelligence they have, demonstrates little flexibility or self-understanding. One of the possible reasons for this is that over the latest years research has become highly specialized: many individuals and research teams focus their efforts on specific problems, narrow in relation to integrated architectures, such as vision, or learning, or language processing, or problem solving, or mobile robotics.

One of the most significant trends in the neuro-technology field over the last decade of the twentieth century has been the development of “biomimetic” electronic systems that emulate the function of biological neural systems. "Biomimetics" refers to technologies that imitate or mimic certain biological functions or even whole organisms. The concept has been used in a bewildering variety of definitions. It is all about examining how biological systems solve problems and then to construct analogous, biomimetic, mechanisms by using reconfigurable hardware and software (Kamat, R., 2010).

# 2 SECOND CHAPTER: PROBLEM DESCRIPTION AND GOALS

## 2.1 Aims and Objectives

The high level aim of this project is to define and evaluate a biologically inspired cognitive architecture model influenced by contemporary research in biomimetics and to develop this as a framework for cognitive control in a real-time embedded system, i.e. a mobile robot.

The following sub-goals are considered necessary for achieving the objectives of the thesis project:

- Perform a survey of what might be meant by cognitive architecture;
- Choose a reference application and find out as much as possible about examples of its robots, e.g. mechanical structure, typical actuators and sensors, control methods;
- Develop a version of a cognitive architecture and apply it to the reference application;
- Evaluate the designed architecture's performance.

## 2.2 Research Questions

The aim of this project is to make some preliminary investigations towards the solution of the following research questions:

**RQ1**: What is the state-of-art in the current research and development of integrated cognitive architectures and how can this be advanced?

**SQ1**: What cognitive architectures have already been developed and are being developed at the moment of writing?
**SQ2**: What is the potential for a cognitive architecture model, more strongly based upon the foundations of biological cognition?

**RQ2**: How can the new framework for an integrated cognitive architecture be applied to a mobile robot?

**SQ4**: What are the necessary characteristics of a mobile robot making it an appropriate platform for the cognitive architecture's implementation? How should it be applied?
**SQ5**: What is the process of implementing the architecture into a mobile robot?

**RQ3**: What are the abilities of a mobile cognitive robot to deal with problems and uncertainty in the task environment?

**SQ6**: What is the robot's ability to process sense data?
**SQ7**: What is the robot's ability to make decisions about how to act in order to satisfy current goals?
**SQ8**: What is the robot's ability to generate behaviours for accomplishing goal-oriented tasks?
**SQ9**: What is the robot's ability to learn?

## 2.3 Outcomes

The project's outcome is a thesis report covering several aspects:

- A comprehensive review of the current "state-of-the-art" in a field of research of cognitive theories and architectures;
- A comparison of current synthetic cognitive architectures based upon or inspired by biological cognitive architectures;
- A list of potential improvements of current architectures based upon this comparison;
- A description of a generic cognitive architecture that explores such proposed improvements, and allows the distribution of tasks between a robot and a real time off-board processing system;
- A list of requirements for a reference application;
- A process guideline describing how the architecture can be implemented into a real-time embedded system;
- Evaluation of the proposed architecture by means of experimental results' evaluation.

# 3 THIRD CHAPTER: RESEARCH METHODOLOGY

The thesis involves different research approaches:

1. Literature study is conducted to gain a fundamental understanding of integrated cognitive architectures in both artificial and biological systems. The same will be applied in order to understand the characteristics of cognitive control in real-time embedded systems.

This approach is suitable since considerable research work is already available related to the study. The literature study embraces existing articles, books and web resources as appropriate. Here, it is noteworthy that a literature study can be time-consuming and therefore, only the major research work shall be considered.

2. Design and implementation of a cognitive architecture.
3. Case study, conducted by investigating the characteristics and abilities of a simple mobile robot in an experimental environment.
4. Experiments in a simulated environment

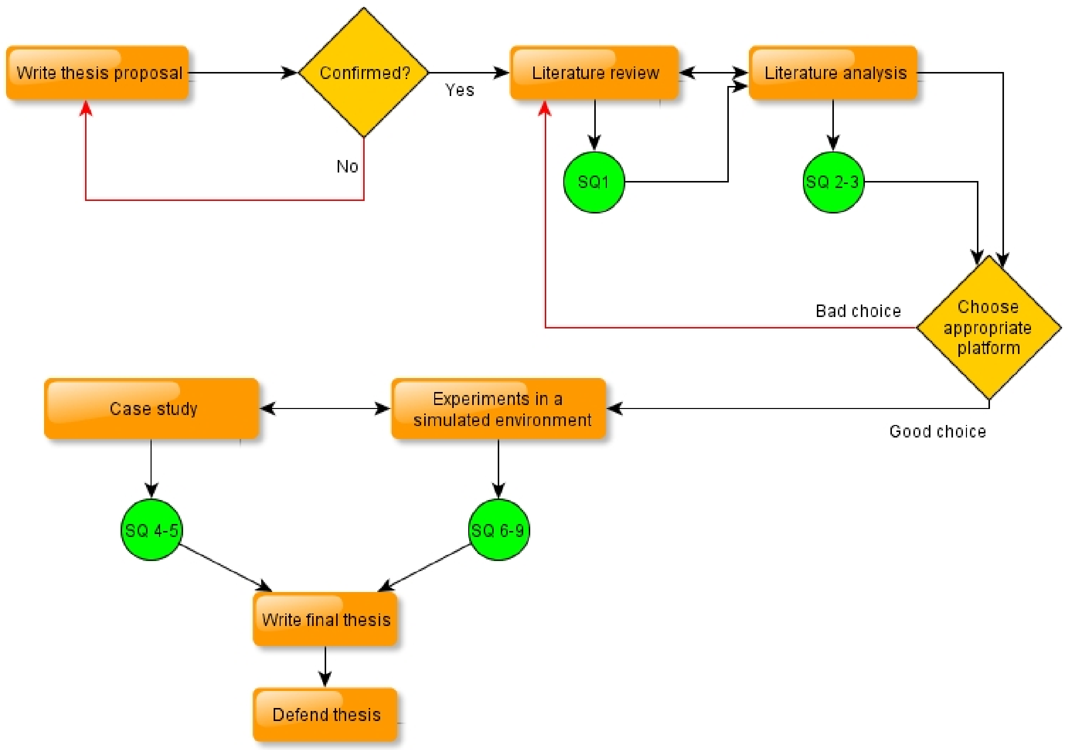
Experiments in a simulated environment are often compared to and contradicted to "real-world" experiments. Some researchers argue that "simulation methods often yield very little in terms of actual theory development. They suggest that simulations are simply "toy models" of actual phenomena, in that they either replicate the obvious or strip away so much realism that they are simply too inaccurate to yield valid theoretical insights" (Chattoe, 1998; Fine and Elsbach, 2000).

However, the author sees this approach suitable, as it has some major advantages over real-world experiments. First of all, simulations are less limited and less expensive, compared to real-world experiments. Simulations are easy to perform even for complex systems, and could be performed in much shorter time, especially when applying genetic algorithms. Moreover, simulations usually have several essential capabilities, such as path creation, accurate motion and cycle time estimation, collision detection and zero collision damage.

5. Evaluating and comparing the results obtained from the experiment.

The results obtained during the experiments are evaluated and compared using several statistical analysis methods, such as one-way and two-way Anova, regression test and trend analysis. These methods are suitable for the research since they present statistically significant comparison of collected data and reveal the correlation or its absence between selected variables.

The research process is schematically described in the picture below:



Picture 1. The process of writing the Master thesis.

# 4 FOURTH CHAPTER: THEORETICAL WORK

## 4.1 Integrated Cognitive Architectures

### 4.1.1 Cognitive Models and Integrated Cognitive Architectures

The idea of a full-featured model of the crucial components of human cognition was advanced by Alan Newell and Herbert Simon as a consequence of the physical symbol system hypothesis (Newell & Simon, 1976). According to this hypothesis, a physical symbol system, that is, an implemented Turing machine, "has the necessary and sufficient means for general intelligent action. By "necessary" we mean that any system that exhibits general intelligence will prove upon analysis to be a physical symbol system. By "sufficient" we mean that any physical symbol system of sufficient size can be organized further to exhibit general intelligence" (Newell, 1987).

A system, which is capable to fulfil the breadth of cognitive tasks required for general intelligence is a model of a unified theory of cognition (Newell, 1987), an implementation of a so-called cognitive architecture (CA) (Joscha Bach, 2009).

An integrated cognitive architecture can be defined as a single system that is capable of producing all aspects of behaviour, while remaining constant across various domains and knowledge bases, thus modelling human performance in multimodal multiple task situations (Newell 1987; Anderson et al. 2003). It's important to notice that this definition of cognition covers not only the brain-mediated functions, but also not-mediated ones, such as peripheral nervous system functions. Another definition of integrated cognitive architecture could define it as a system, integrating across a variety of functional modules (Wayne, 2007). So, a cognitive architecture includes the "foundations" of the cognitive system, the divisions of various modules, essential relations between modules, basic representations and algorithms within modules, and a variety of other aspects (D. J. Jilk et al., 2008). Cognitive architecture research seeks to define a collection of integrated processes and knowledge representations that provide a general foundation for cognitive behaviour across an increasingly broad spectrum of problems (Duch, W. et al., 2008).

The goal of building cognitive architectures is to achieve an understanding of mental processes by constructing testable information processing models. Every implementation that does not work, that is, does not live up to the specifications that it is meant to fulfil, points out gaps in understanding (Joscha Bach, 2009).

For an artificial cognitive system (ACS) to be used in complex environments it should be capable of responding intelligently and autonomously to gaps in its knowledge and to contexts that have not been specified in the initial design (Buiu, C., 2009). Additionally, ACSs ought to be more effective in improving their performance by learning from their actions, and more natural ways to interact with such systems are needed to extend the range of possible users and ease the interaction with the system (Buiu, C., 2009).

A CA should capture the essential structure and process of the mind, to be used for a broad, multiple-level, multiple-domain analysis of behaviour (Newell, A., 1990). According to Sun (2007), the function of a CA is to provide an essential framework to facilitate modelling and exploration of components and various processes of human mind.

### 4.1.2 Classification of Integrated Cognitive Architectures

Architectures, in general, have divergent features that lead to different properties, they support specific capabilities and are used in different domains and environments. All these differences are responsible for the variety of architectures. Some authors (Joscha Bach, 2009) divide cognitive architectures into the following methodological groups:

- Classical (symbolic) architectures, which are essentially rule-based. These sprang up after Newell's call for a revival of unified theories in psychology (Newell, 1973, 1987). Classical architectures concentrate on symbolic reasoning, bear influences of a relatively strict language of thought concept, as suggested by Fodor, and are often implemented as production based language interpreters. Gradually, these architectures have been modified to allow for concept retrieval by spreading activation, the formation of networks from the initial rules and have occasionally even been implemented based on neural elements (Joscha Bach, 2009).

- Parallel distributed processing (PDP) (subsymbolic) architectures. This term was introduced by James McClelland (Rumelhart, McClelland et al., 1986); here, it is used to refer to nonsymbolic distributed computing (usually based on some or several types of recurrent neural networks). Where classical architectures strive to attain the necessary complexity by carefully adding computational mechanisms, PDP systems are inspired by biological neural systems. Their contemporary forms essentially work by constraining a chaotic system enough to elicit orderly behaviour. While PDP architectures do not necessarily differ in computational power from classical architectures, it is difficult to train them to perform symbolic calculations, which seem to be crucial for language and planning. On the other hand, they seem to be a very productive paradigm to model motor control and many perceptual processes (Joscha Bach, 2009).

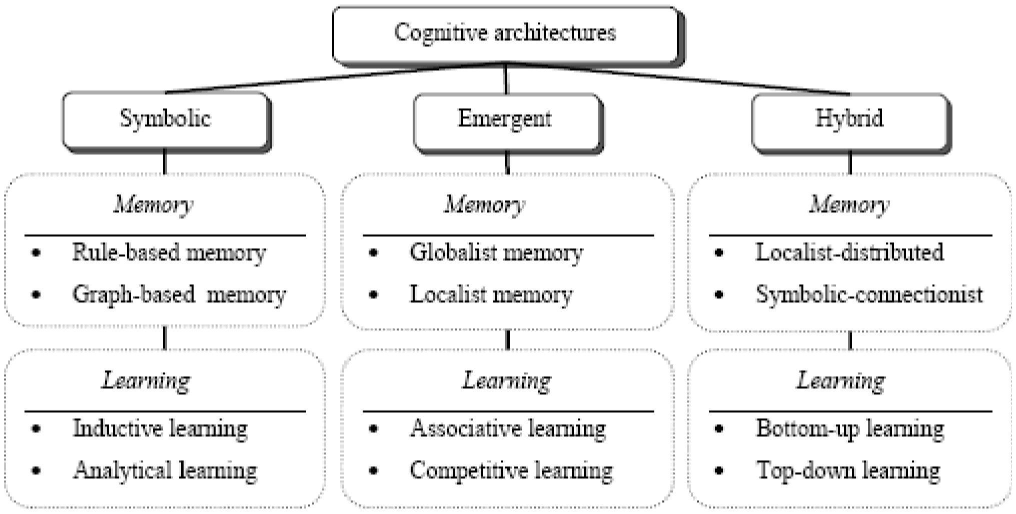
- Hybrid architectures, which use different layers for different tasks: a reasoning layer that performs rule-based calculations, and a distributed layer to learn and execute sensory-motor operations. Hybrid architectures are usually heterogeneous (i.e., they consist of different and incompatible representational and computational paradigms that communicate with each other through a dedicated interface), or they could be homogenous (using a single mode of representation for different tasks). The latter group represents a convergence of classical and PDP architectures, and our own approach follows this direction (Joscha Bach, 2009).

- Biologically inspired architectures, which try to directly mimic neural hardware—either for a complete (simple) organism or as a layer within a hybrid approach (Joscha Bach, 2009).

- Emotional and motivational architectures. Emotion and motivation are vital parts of a cognitive system, but this distinction does not take care of how they are introduced into the system. This is because most existing models either ignore them or treat them as separate entities, situated and discussed outside the core of the model. Exceptions to the rule exist, of course, for instance Clarion (Sun, 2003, 2005), PURR-PUSS (Andreae, 1998) and of course the PSI theory, which all treat emotion and motivation as integral aspects of the cognitive system. For many other cognitive architectures separate additions exist, which provide an emotional or motivational module that interfaces with the cognitive system (Belavkin, Ritter, & Elliman, 1999; Norling & Ritter, 2004; Franceschini, McBride, & Sheldon, 2001; Gratch & Marsella, 2001; Jones, 1998; Rosenbloom, 1998).

Others (Duch W. et al, 2008) group cognitive architectures according to the features of memory organization and knowledge representation schemes into three main groups:

- Symbolic. Roughly speaking symbolic architectures focus on information processing using high-level symbols or declarative knowledge, in a classical AI top-down, analytic approach.
- Emergent. Emergent architectures use low-level activation signals flowing through a network consisting of numerous processing units, a bottom-up process relaying on the emergent self-organizing and associative properties.
- Hybrid models. Hybrid architectures result from combining the symbolic and emergent paradigms in one way or another.

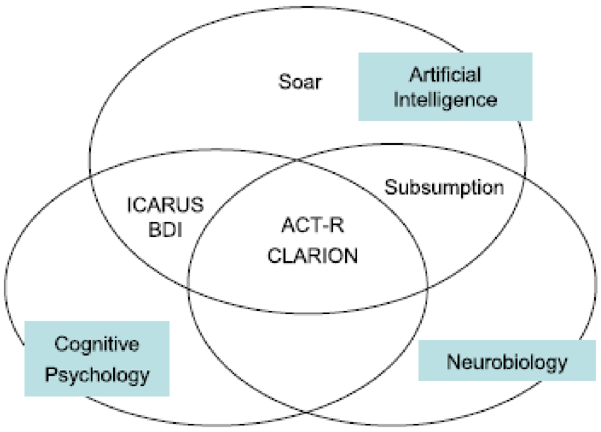


Picture 2. Simplified taxonomy of cognitive architectures (Duch W. et al, 2008)

Others, as for example Sun (2006), distinguish between:

- psychologically oriented and
- software engineering oriented CAs.

Figure 3 shows six cognitive architectures, namely Soar, ACT-R, ICARUS, BDI, the subsumption architecture, and connectionist learning with adaptive rule induction on-line (CLARION), roughly classified according to their roots and emphases (Hui-Qing Chong et al., 2009).



Picture 3. The overview of the six cognitive architectures (Hui-Qing Chong et al., 2009)

Over the past decades, many cognitive architectures have been proposed and steadily developed, based on different approaches and methodologies, such as Soar (Laird et al. 1986a, b; Lehman et al. 2006), ACT-R (Anderson et al., 2004), ICARUS (Langley and Choi 2006), BDI (Bratman et al. 1988; Rao and Georgeff 1991), CLARION (Sun and Peterson 1996, 1998; Sun et al. 2001; Sun and Zhang 2006) and others.

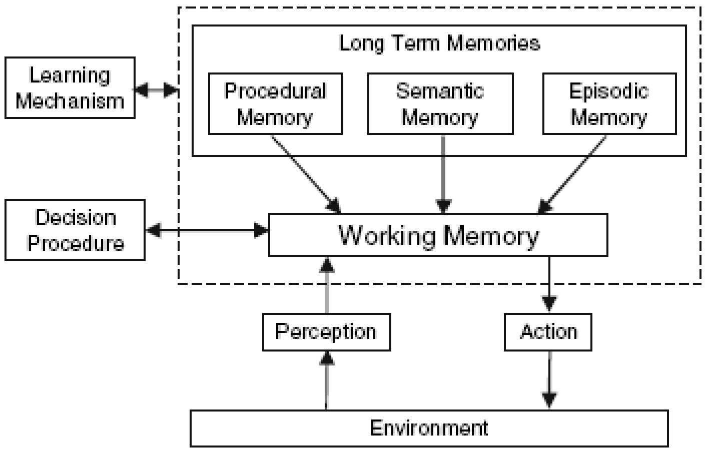
### 4.1.3 Review of Integrated Cognitive Architectures

#### 4.1.3.1 Symbolic Architectures

A venerable tradition in artificial intelligence (AI) focuses on the physical symbol system hypothesis [2], which states that minds exist mainly to manipulate symbols that represent aspects of the world or themselves. Generally, symbolic cognitive architectures focus on ''working memory'' that draws on long-term memory as needed, and utilize a centralized control over perception, cognition and action.

A few of the best known symbolic cognitive architectures are as follows:

- ***SOAR***. Soar (State, Operator And Result) is one of the first cognitive architectures. It was designed as an expert rule-based system, intended to model general intelligence through a mechanism of learning from experience.

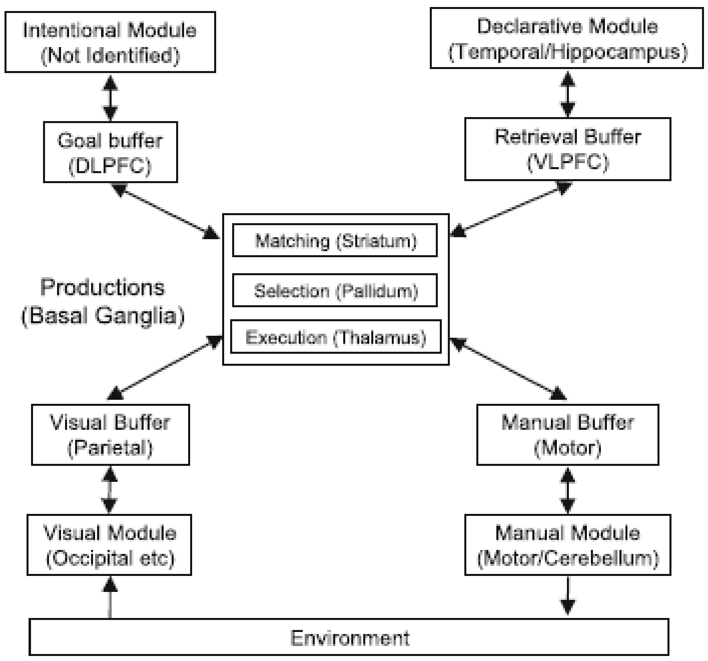


Picture 4. Soar integrated cognitive architecture (Hui-Qing Chong et al., 2009).

The primary principle of the Soar design is the statement that “all decisions are made through the combination of relevant knowledge at run-time. In Soar, every decision is based on the current interpretation of sensory data, the contents of working memory created by prior problem solving, and any relevant knowledge retrieved from long-term memory. Decisions are never precompiled into uninterruptible sequences.” [http://sitemaker.umich.edu/soar/home]. This architecture is not yet strong in such areas as creativity, handling uncertain knowledge and reinforcement learning.

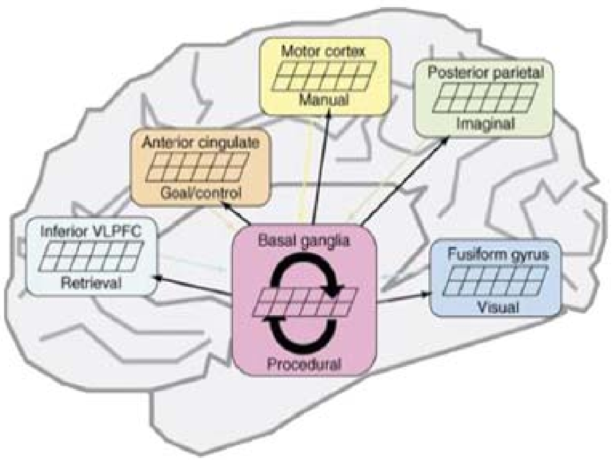
Soar has a relatively large user base among existing cognitive architectures. It is supported by the University of Michigan and has been applied commercially by Soar Technology Inc.

- ***ACT-R***. This architecture is founded on the knowledge, gained from the experiments in cognitive psychology and brain imaging.



Picture 5. ACT-R integrated cognitive architecture (Hui-Qing Chong et al., 2009).

The picture below shows the association between the modules in ACT-R architecture and human brain regions:

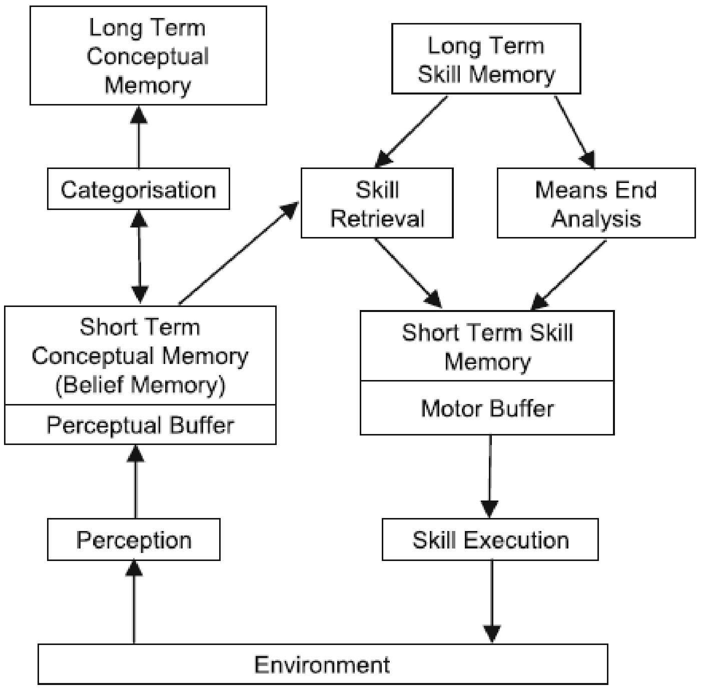


Picture 6. How the various cortical modules of ACT-R are coordinated through the procedural module that is associated with the basal ganglia (Goertzel, B. et al., 2010).

ACT-R (Adaptive Control of Thought-Rational) integrates multiple modules that correspond to different cognitive functions. The fundamental controlling structure in cognition is reactive–where production rules respond to patterns of information in various cognitive buffers. This model assumes a mixture of parallel and serial processing. Parallelism occurs within each module as well as between different modules. There are two levels of serial bottleneck in this model – limited knowledge space in every buffer and only single production in every cycle.

The ACT-R architecture is not very strong in such areas as abstract reasoning, creativity, and transfer learning.

- ***ICARUS***. This is an integrated cognitive architecture for intelligent agents where knowledge is specified in the form of reactive skills, each denoting goal-relevant reactions to a class of problems.



Picture 7. ICARUS integrated cognitive architecture (Hui-Qing Chong et al., 2009).

The key processes in ICARUS include conceptual inference, goal selection, skill execution, problem solving, and learning (Langley and Choi 2006). The design for ICARUS has been guided by the following principles, which provide the differentiation of ICARUS from other cognitive architectures (Hui-Qing Chong et al., 2009):

- ✓ Cognitive reality of physical objects;
- ✓ Cognitive separation of categories and skills;
- ✓ Primacy of categorization and skill execution;
- ✓ Hierarchical organization of long term memory;
- ✓ Correspondence of long term or short term structure;
- ✓ Modulation of symbolic structures with utility functions.

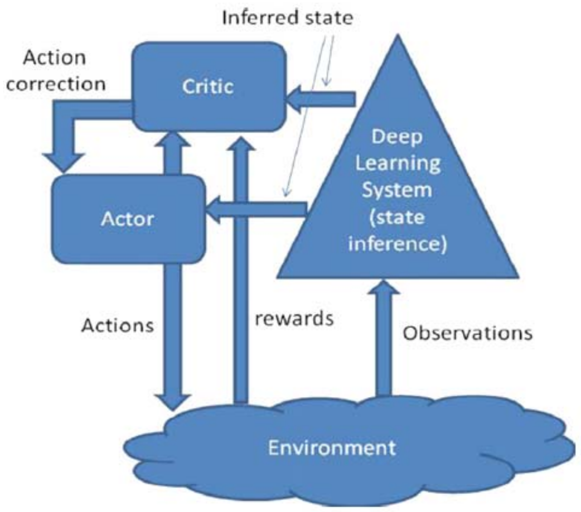
ICARUS is a general purpose architecture which is not yet broadly supported. Potential applications to integrated cognition remain to be determined. Concurrent processing is absent, attention allocation is fairly crude, and uncertain knowledge is not thoroughly handled.

Although in principle symbolic architectures could be arbitrarily capable (since symbolic systems have universal representational and computational power, in theory), in practice symbolic architectures tend to be weak in learning, creativity, procedure learning, and episodic and associative memory (Goertzel, B. et al., 2010).

#### 4.1.3.2 Subsymbolic Architectures

Another species of cognitive architecture expects abstract symbolic processing to emerge from lower-level ’’subsymbolic’’ dynamics, which usually (but not always) are heavily biologically inspired, and designed to simulate neural networks or other aspects of human brain function.

- ***DeSTIN***. This is a hierarchical system, which contains hierarchical perception network and coordinated hierarchical networks dealing with action and reinforcement:



Picture 8. DeSTIN integrated cognitive architecture (Goertzel, B. et al., 2010).

The DeSTIN architecture is going away from the “shallow” structures of machine learning and pattern recognition systems and following the approach of hierarchical structure of the world with the brain-like “deep learning”. The spatiotemporal network of DeSTIN has three layers:

✓ at the very lowest layer of the hierarchy nodes receive as input raw data (e.g. pixels of an image) and continuously construct a belief state that attempts to characterize the sequences of patterns viewed,

✓ the second layer, and all those above it, receive as input the belief states of nodes at their corresponding lower layers, and attempt to construct belief states that capture regularities in their inputs, and

✓ each node also receives as input the belief state of the node above it in the hierarchy (which constitutes “contextual” information) (Goertzel, B. et al., 2010).

- ***NOMAD***. <u>N</u>eurally <u>O</u>rganized <u>M</u>obile <u>A</u>daptive <u>D</u>evice automata are based on “neural Darwinism” theory (Edelman, G., 1993). Nomads, also known as Darwin automata, demonstrate the principles of emergent architectures for pattern recognition task in real time.

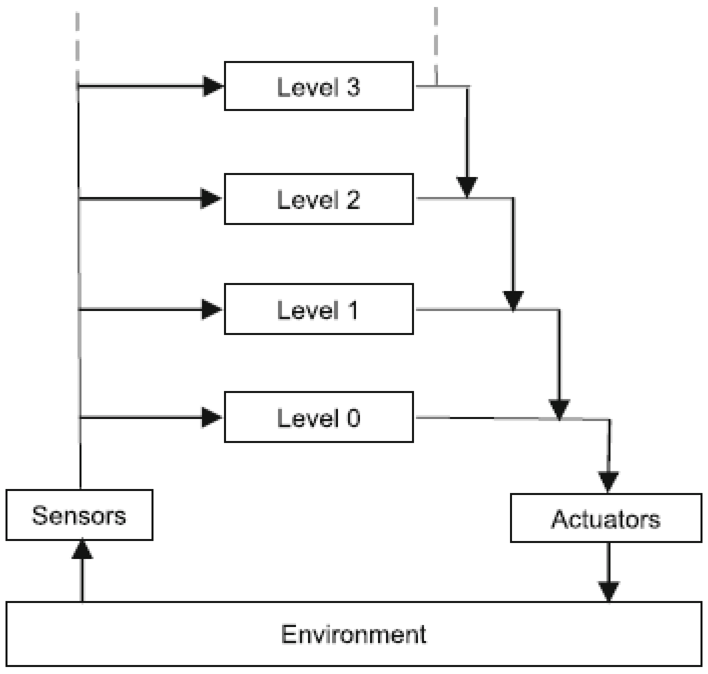
The main principles of NOMAD device are:

- ✓ the device must engage in a behavioural task;
- ✓ the device's behaviour must be controlled by a simulated nervous system having a design that reflects the brain's architecture ad dynamics;
- ✓ the device's behaviour is modified by a reward or value system that signals the salience of environmental cues to its nervous system;
- ✓ the device must be situated in the real world.

Darwin models use many sensors for vision, range finders to provide a sense of proximity, prioproceptive sense of head direction and self-movement, artificial whiskers for texture sensing, artificial taste (conductivity) sensors. NOMAD is controlled by a very large (~$10^5$ neurons with ~$10^7$ synapses) simulated nervous system with 28 brain areas modelled.

However, the emergence of higher-level cognition does not seem likely in this architecture.

- ***Subsumption architecture***. One of the best examples of reactive architectures is a subsumption architecture (Brooks, 1999). A subsumption architecture is a way of decomposing complicated intelligent behaviour into many "simple" behaviour modules, which are in turn organized into layers. Each layer implements a particular goal of the agent, and higher layers are increasingly abstract. Each layer's goal subsumes that of the underlying layers, e.g. the decision to move forward by the eat-food layer takes into account the decision of the lowest obstacle-avoidance layer. As opposed to more traditional AI approaches subsumption architecture uses a bottom-up design.



Picture 9. Subsumption integrated cognitive architecture (Hui-Qing Chong et al., 2009).

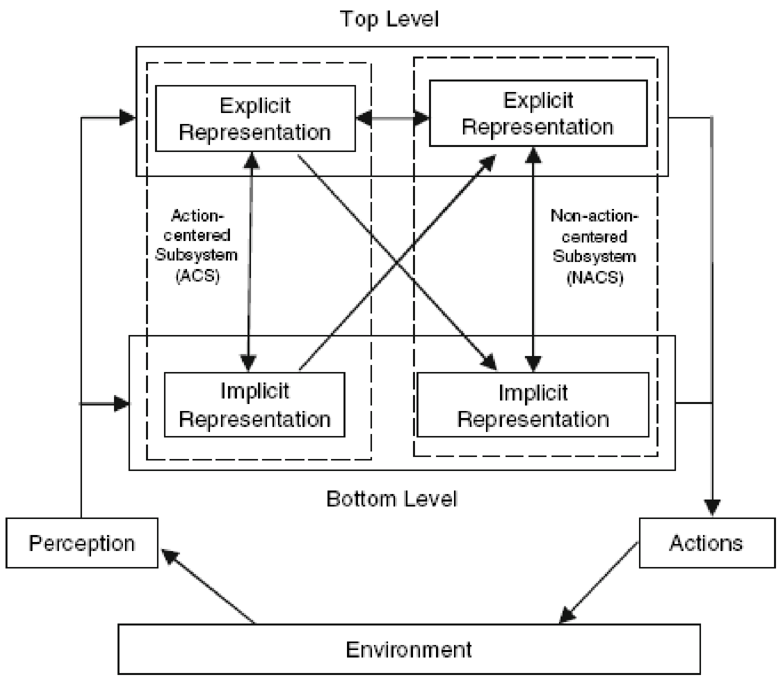
The subsumption architecture is able to cope with noisy sensory information and unpredictable or volatile environment. It is free from central control, but the higher layers can suppress the inputs and inhibit the outputs of the lower layers, leading to an adjustment in the behaviour for the purpose of fulfilling the overall goal (Butler et al. 2001; Brooks 1999; Amir and Maynard-Zhang 2004).

The subsymbolic architectures are typically strong at recognizing patterns in high-dimensional data, reinforcement learning and associative memory; but no one has yet shown how to achieve high-level functions such as abstract reasoning or complex language processing using a purely subsymbolic approach (Goertzel, B. et al., 2010).

#### 4.1.3.3 Hybrid Architectures

Finally, in response to the complementary strengths and weaknesses of the symbolic and emergentist approaches, in recent years a number of researchers have turned to integrative, hybrid architectures, which combine subsystems operating according to the two different paradigms. The combination may be done in many different ways, e.g. connection of a large symbolic subsystem with a large subsymbolic system, or the creation of a population of small agents each of which is both symbolic and subsymbolic in nature (Goertzel, B. et al., 2010).

- ***Clarion***. Connectionist learning with adaptive *r*ule *i*nduction on-line is a hybrid cognitive architecture that incorporates both implicit (procedural) and explicit (declarative) types of knowledge for reasoning and learning. The architecture is composed of four main subsystems: the Action-centred Subsystem, the Non-action-centred Subsystem, the Motivational Subsystem and the Meta-cognitive Subsystem.

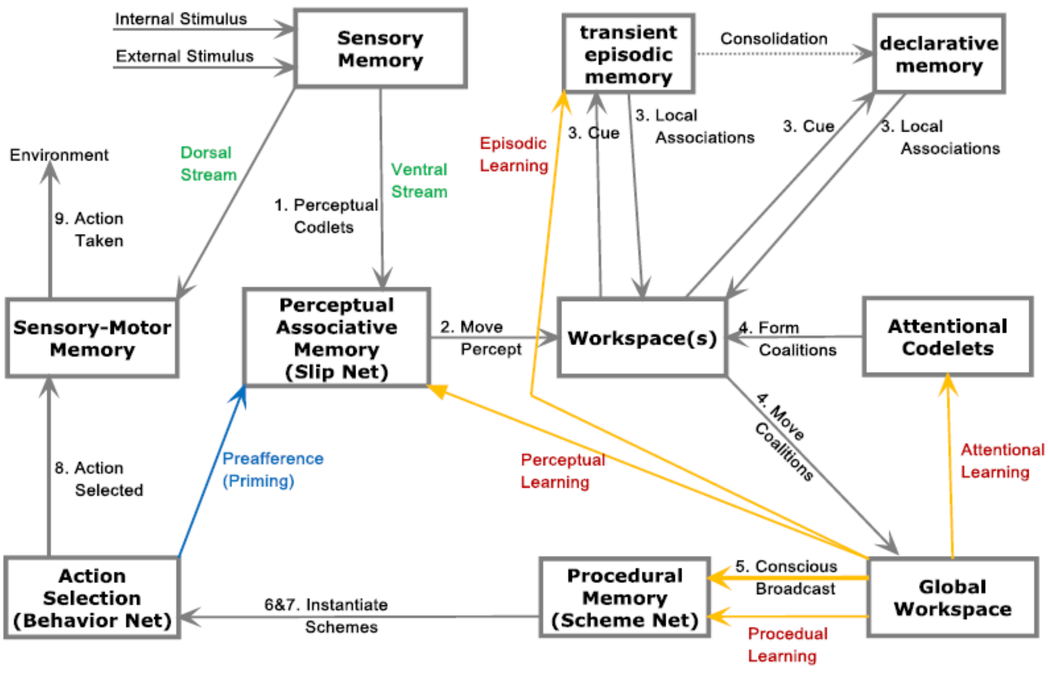


Picture 10. Clarion integrated cognitive architecture (Hui-Qing Chong et al., 2009).

The representational difference between the two types of knowledge, relates to accessibility. In each subsystem of the architecture, the top level contains easily accessible explicit knowledge whereas the bottom level contains less accessible implicit knowledge.

The bottom-up learning is another specific characteristic of Clarion architecture. In each subsystem, implicit associations are learned in the bottom level through trial-and error learning. However, in the bottom level learning of explicit knowledge is one-shot [6]. Learning of procedural knowledge at the bottom level occurs through the reinforcement learning paradigm, which works by making adjustments to the probability of selecting a particular action (Hui-Qing Chong et al., 2009).

- ***LIDA***. The Learning Intelligent Distribution Agent architecture was developed by Stan Franclin and his colleagues (D. Friedlander, S. Franklin, 2008) as an attempt to model a broad spectrum of cognition in biological systems. Two hypotheses underlie the LIDA architecture and its corresponding conceptual model: 1) Much of human cognition functions by means of frequently iterated (~10 Hz) interactions, called cognitive cycles, between conscious contents, the various memory systems and action selection. 2) These cognitive cycles, serve as the "atoms" of cognition of which higher-level cognitive processes are composed (http://en.wikipedia.org/wiki/LIDA_(cognitive_architecture)).



Picture 11. LIDA cognitive cycle (Goertzel, B. et al., 2010).

The architecture ties in well with both neuroscience and cognitive psychology, but it deals most thoroughly with ''lower level''' aspects of intelligence. The aspects of higher-level cognition, such as language and reasoning, are only sketchily presented in this architecture.

- ***Shruti***. This is an architecture developed by Lokendra Shastri of UC Berkeley [SHAST 1999]. The architecture demonstrates how simple, neuron-like, elements can encode a large body of relational causal knowledge and provide a basis for reactive, rapid inference. Shruti is a biologically-inspired model of human reflexive inference, represents in connectionist architecture relations, types, entities and causal rules using focal-clusters (Goertzel, B. et al., 2010). However, until now this architecture has not proved significant effective in any applications, so for now it cannot be considered as a practical artificial brain system.

### 4.1.4 Bio-inspired Integrated Cognitive Architecture

Biological systems are an important source of information on how to design and implement CAs.

There are several possible approaches for BICA, including:

1) Cognitive architectures:

a. With added abstract biologically-inspired functions.

Such a CA can have an abstract functionality e.g., episodic memory, semantic memory, reinforcement learning, emotions. But this alone will not give it a direct connection to underlying computational process.

b. With added perceptual-motor inputs and outputs.

Such a CA can provide a brain-based computation of perception and action control, but the whole architecture remains essentially unchanged.

c. With biologically-inspired module implementation.

Such a CA uses brain-based computation for existing CA components. The overall structure of CA is retained. Components of such a CA interact in both symbolic and non-symbolic ways.

2) Biological architectures, cognitively constrained.

Such a CA is a new biologically derived architecture, in which cognitive functions emerge from the interaction of existing components.

A lot of work in biologically-inspired cognitive architectures is focused on using neural networks. One of the limitations of neural inspired CAs that have been developed so far is that they tend to solve modal problems (visual recognition of objects, audition, motivation etc.) in disparate architectures whose design is very specialized for each modal problem (D. J. Jilk et al., 2008). There are very often quite different theories and architectures for the same modal cognitive problem. This raises a significant challenge in order to integrate all these disparate theories so that all the desired cognitive abilities are manifested. Computational design and implementation of such an integrated architecture is an important challenge (Newell, A., 1990).

## 4.2 Genetic Programming

Evolutionary algorithms mimic aspects of natural evolution to optimize a solution towards a defined goal. Darwin's principle of natural selection plays a key role when differential fitness advantages are exploited to lead to better solutions.

A general evolutionary algorithm may be summarized as follows:

1. A random population of solutions is created.
2. In an iteration loop: Individuals are selected from the population randomly and are compared, based on their fitness. The fitness measure defines the task which should be reached by the algorithm, or the problem which should be solved. Only the best (fitter) individuals are modified by the following genetic operations while the population size is kept constant.

   The following genetic operations are usually used:
   - Identical reproduction
   - Exchange of a substructure in an individual at a random position (mutation)
   - Exchange of substructures between two individuals (crossover)

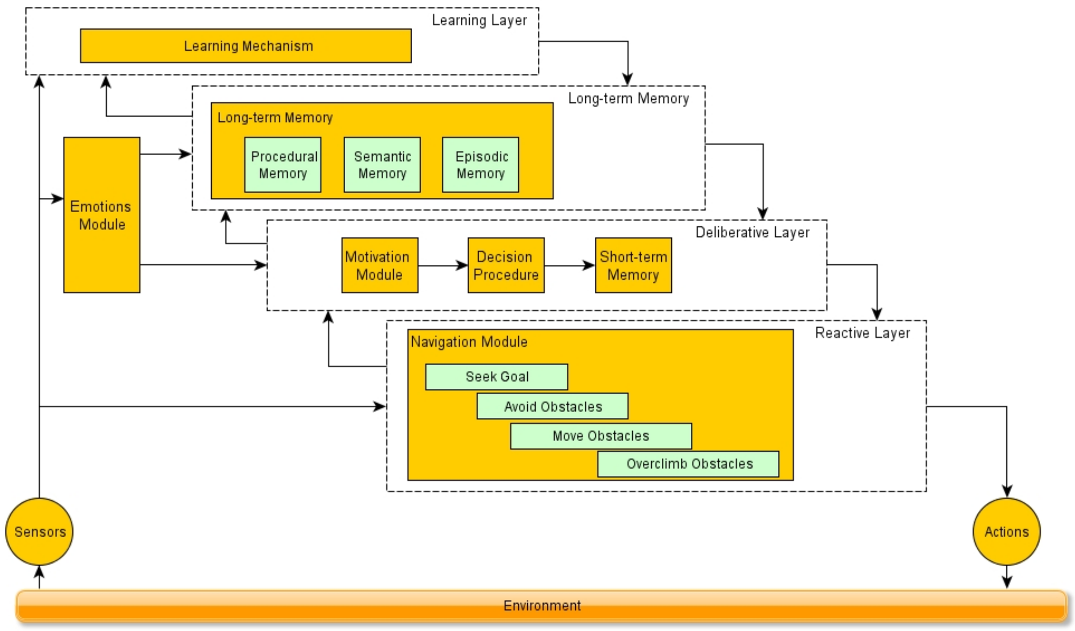
3. The currently best individual in the population represents the best solution found so far.

As genetic programming operates on digital chromosomes of variable size (Mitchell, 1992), it overcomes some of the limitations of genetic algorithms, such as, for example, the necessity to use fixed-length chromosomes, the difficulty in representing hierarchical structures, and the lack of dynamic variability (Koza, 1992).

Genetic programming could be a good learning mechanism in an integrated cognitive architecture. It allows a computer program to evolve constantly and makes the integrated cognitive architecture a dynamic and flexible system. Moreover, earlier research (Koza, 1993) showed that it is possible to integrate genetic programming with cognitive architectures, such as a subsumption architecture.

## 4.3 Proposed Integrated Cognitive Architecture

The architecture presented in the below picture, is proposed by the author as a generic integrated cognitive architecture:



Picture 12. The proposed generic cognitive architecture.

This is hybrid architecture, composed of four main layers – reactive, deliberative, long-term memory and learning layer. It compounds a symbolic and behaviour-based way of organizing and structuring the data. This architecture is biologically inspired: first of all, the navigation module represents a reactive subsumption architecture, which is bio-inspired, and the second, the structure of a long-term memory module, consisting of procedural, semantic and episodic memories, corresponds to a human brain and memory structure.

Operation at a Reactive level is characterized by quick responses (reactions) to sensory or other information that may be processed to varying degrees into representations of world situations. The Reactive level does not exclude formation and use of world or environmental models to guide behaviour, so long as their use does not become deliberative. In my architecture, navigation tasks are primarily reactive, thus the navigation of a robot is fast (Rolfe, R., & Haugh, B., 2003).

Distinguishing Deliberative from Reactive levels is important because deliberative consideration of alternative futures enables cognitive agents to solve more difficult problems than merely instinctive or reactive agents (Rolfe, R., & Haugh, B., 2003). The modules of emotions and motivation extend the capability of a robot to self-reflection and self-consciousness. Self-reflective reasoning can improve an agent’s functioning by enabling it to step back from a difficult problem or goal and reflect upon how it relates to other aspects of itself. Current goals are often derivative of broader interests that may be better served by changing or revising goals when they are frustrated by circumstances (Rolfe, R., & Haugh, B., 2003). Self-consciousness, from the other side, has a tight connection with the emotional module of the architecture. Different self-conscious emotions, such as shame, pride, guilt or envy, are emerging from subjective experience of being, and are causing the self-evaluation, which, as an outcome, is influencing the behaviour. The Self-Conscious level operates on all representations of the self and the world, and thus will result in behaviour that includes cognition, affect, and motivation.

The long-term memory layer in this architecture corresponds to a reflective level of integrated cognition. Functioning at the Reflective level entails an agent representing and reasoning about what it has done and thought. The mind forms representations of its own action, thoughts, and deliberations, as well as the external world.

Learning strategies of artificial cognitive systems are still quite primitive compared to human learning. However, fully capable integrated cognition systems should be able to utilize all forms of learning. The Association for the Advancement of Artificial Intelligence (AAAI) website on machine learning states that machine learning is said to occur in a program that can modify some aspect of itself, often referred to as its state, so that on a subsequent execution with the same input, a different (hopefully better) output is produced (AAAI 2004). Different strategies for learning may be distinguished by the amount and types of inference required of the learner during the course of learning. Simply adding data to system information stores can increase its knowledge, but this would be considered a very simple learning strategy. In contrast, a system capable of performing experiments and generalizing the results to new scientific theories exhibits a very sophisticated learning strategy. Higher forms of learning may depend upon the appropriate level of unification to be included in the cognitive system, e.g., creativity (Rolfe, R., & Haugh, B., 2003).

## 4.4 Mobile autonomous robots

### 4.4.1 Requirements for a cognitive research robot

Physical robots are usually large and fragile. The below table presents lengths, widths and weights of some robots, used for research and education:

| | **Seekur®** | **Seekur Jr** | **PowerBot™** | **Research PatrolBot®** | **Research GuiaBot®** | **PeopleBot™** | **Pioneer 3-DX** | **Pioneer 3-AT** | **AmigoBot TM** |
|---|---|---|---|---|---|---|---|---|---|
| Length (cm [in]) | 140 [55] | 105 [41] | 85 [33] | 59 [23] | 59 [23] | 47 [19] | 45 [18] | 50 [20] | 33 [13] |
| Width (cm [in]) | 130 [51] | 84 [33] | 63 [25] | 48 [19] | 48 [19] | 38 [15] | 40 [16] | 49 [19] | 28 [11] |
| Height without accessories (cm [in]) | 110 [43] | 50 [20] | 47 [19] | 38 [15] | 131 [52] | 112 [44] | 24.5 [9.5] | 26 [10] | 15 [6] |
| Ground Clearance (cm) | 18 [7.1] | 10 [3.9] | 6.7 [2.6] | 5 [2.0] | 5 [2.0] | 3.5 [1.4] | 6.5 [2.5] | 7 [2.8] | 3 [1.2] |
| Weight (with min. battery capacity) (kg [lbs]) | 300 [660] | 80 [176] | 120 [264] | 46 [101] | 54 [119] | 12 [26] | 9 [20] | 12 [26] | 3.6 [8] |

Table 1. (Adopted from http://www.mobilerobots.com)

Therefore, they require large dedicated spaces and could be badly damaged by bio-inspired controllers that go through an adaptation phase based on trials and errors. From this perspective, mobile robots to be used for research and development of bio-inspired adaptive systems should be robust, relatively compact, reliable, easily interfaced with standard computer platforms and software, and equipped with solutions for power supply and behavioural monitoring.

#### 4.4.1.1 Miniaturization

Robot miniaturization can bring several advantages, if realized in the appropriate proportions:

- the experimenter can build complex environments on a limited surface,
- compact working surface also allows an easier monitoring of the robot behaviour,
- fundamental laws of physics give higher mechanical robustness to a miniature robot.

However, miniaturisation can also bring some drawbacks, such as the difficulty of mounting large devices on the robot (ultrasonic sensors, laser range finders, etc.).

#### 4.4.1.2 Modular open architecture

A modular open architecture enables different possible configurations and experiments using the same basic components. It also means possibilities for extensions that are developed for specific research goals, such as bio-morphic sensors or actuators.

#### 4.4.1.3 Accessibility and cross platform compatibility

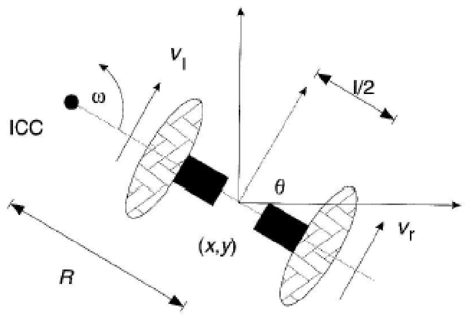
There should be a possibility to attach a cognitive research robot to any computer through a standard connection and contacts. The connection should support fast data communication, as some experiments require extended periods of time (such as an evolutionary approach to learning, for example). Cross platform compatibility has to ensure the possibility to transfer tested algorithms from one robot to another. Cross platform compatibility in mobile robotics is a rare feature, especially between robots of very different sizes. Software compatibility alone is not sufficient to guarantee a good transfer of algorithms between different robots. Mechanical structure, sensor characteristics, and many other aspects of the physical platform also play an important role in the feasibility of the transfer.

One of the robots that correspond to most of the criteria mentioned above is the Khepera robot, which has been designed on the basis of the following criteria: miniaturization, modular open architecture, expandability, interface, and compatibility with larger robots.

### 4.4.2 Khepera mobile robot

Khepera is a miniature mobile robot with functionality similar to that of larger robots used in research and education. Khepera was originally designed as a research and teaching tool for a Swiss Research Priority Program at EPFL in Lausanne. It allows real world testing of algorithms developed in simulation for trajectory planning, obstacle avoidance, pre-processing of sensory information, and hypotheses on behaviour processing, among others.

The motor system of Khepera robot uses two lateral wheels and supporting pivots in the front and back. This configuration allows rotation of the body without lateral displacement. Thus, Khepera is a differential robot with two wheels on a common axis, where each wheel can be independently driven either forward or backward. By varying the velocity of the two wheels, the trajectories the robot takes can be varied:



Picture 13. Khepera model (Adopted from Dudek and Jenkin, 2000).

$R = l/2 * (v_l + v_r)/(v_r - v_l)$

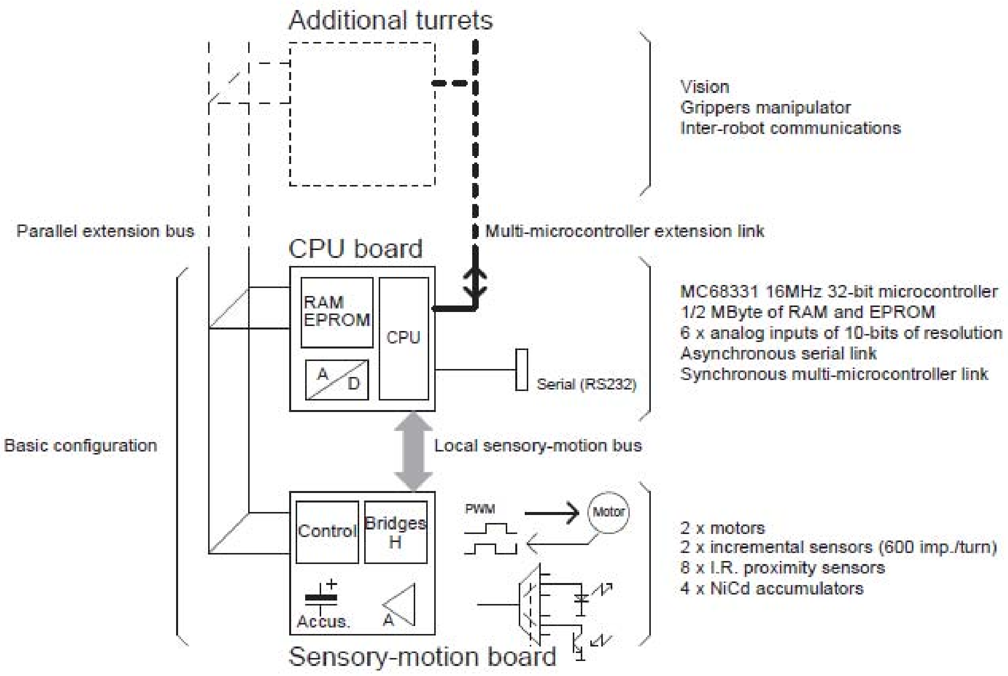
Here
l - is the distance between the two wheels
R - is the radius of the ICC (Instantaneous Centre of Curvature)
$v_r$ - is the velocity of the right wheels
$v_l$ - is the velocity of the left wheels.

The sensory system uses eight active infrared-light sensors distributed around the body, six on one side and two on the other (this asymmetry can be used to establish the front and back of the robot). These sensors can detect the presence of objects by emitting and measuring reflected light (the closer the object, the stronger the response), but can also be used to measure the infrared component of ambient light. Four rechargeable NiCd batteries with a total autonomy of approximately 30-40 minutes are secured on the sensorimotor board. The CPU board includes the robot's main processor (a Motorola MC68331 with 128 Kbytes of EEPROM and 256 Kbytes of static RAM), an A/D converter for the acquisition of analogue signals coming from the sensorimotor board, and an RS232 serial-line miniature connector which can support data transmission and power supply from an external computer.

Picture 14. The first generation Khepera robot, Khepera II and Khepera III.

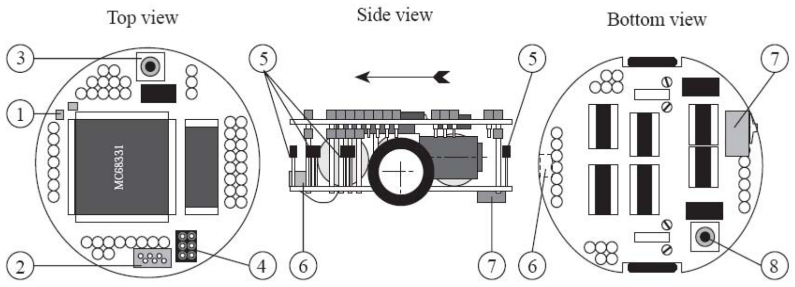
a) Hardware architecture

Khepera has an extension bus that makes it possible to add turrets on the top of the basic configuration, depending on the needs of the experiments that one wishes to perform. This modularity is based on a parallel and a serial bus. The parallel bus can be used for simple extensions directly under control of the main Khepera processor. The serial bus implements a local network for inter-processor communication. Using this second bus, other processors can be connected to the main one in order to build a multi-processor structure centred on the Khepera main processor. This kind of structure has the advantage that one can employ additional computational devices on extension modules, thus keeping the main processor free for global management of the robot behaviour.


Additional turrets
Vision
Grippers manipulator
Inter-robot communications
Parallel extension bus
Multi-microcontroller extension link
CPU board
RAM
EPROM
CPU
A
D
Serial (RS232)
MC68331 16MHz 32-bit microcontroller
1/2 MByte of RAM and EPROM
6 x analog inputs of 10-bits of resolution
Asynchronous serial link
Synchronous multi-microcontroller link
Basic configuration
Local sensory-motion bus
Control
Bridges H
PWM
Motor
Accus.
A
2 x motors
2 x incremental sensors (600 imp./turn)
8 x I.R. proximity sensors
4 x NiCd accumulators
Sensory-motion board

Picture 15. Khepera hardware architecture and extensions (K-Team, 1999)

b) Mechanical structure



Picture 16. Position of some parts of the robot (K-Team, 1999).

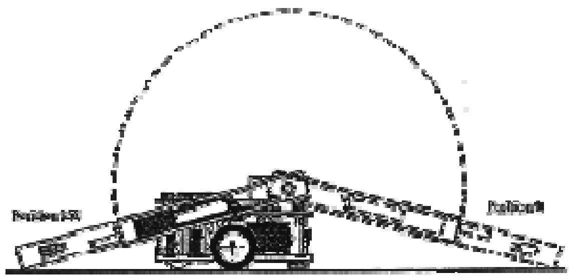
Here the numbers correspond to the following parts of the robot:
1. LEDs
2. Serial line (S) connector.
3. Reset button.
4. Jumpers for the running mode selection.
5. Infra-Red proximity sensors.
6. Battery recharge connector.
7. ON - OFF battery switch.
8. Second reset button (same function as 3).

The ROM installed on this robot has an important library of software modules for the real time control of the Khepera robot. A subset of these modules provides the basic functionalities of the Khepera robot, such as motor control, sensor scanning, etc.. Another subset of these modules provides the interface with the user through the serial line. Depending on the use of the robot (remote control, downloading, test, demo etc.) it’s possible to select a specific module by setting the correspondent running mode.

c) Typical actuators

***Gripper Module***

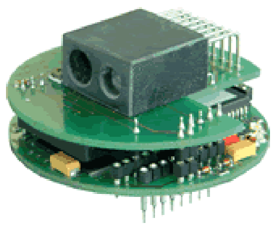
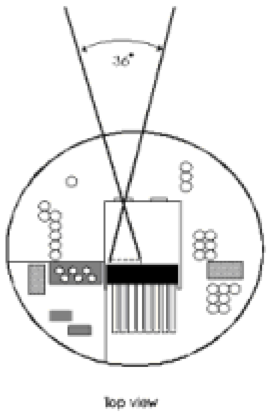
- Built-in CPU for local control
- Detects objects between the gripper fingers
- Able to examine electric resistance of the object in the gripper



Picture 17. Gripper turret

***K213 Vision Turret***

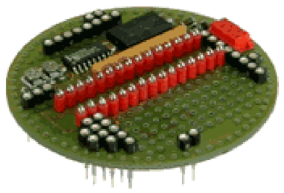
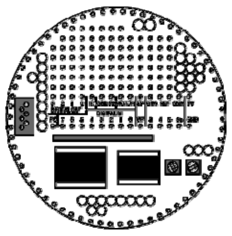
- Linear vision (1 x 64 pixels)
- 256-level grey scale
- Automatic iris adjustment
- Light intensity detection
- On board pre-processing

Picture 18. Vision Turret

***General I/O Turret***

- Framework to mount application-specific sensors and actuators
- Digital input/output
- Variable gain analogue input

Picture 19. I/O turret.

***IR Communication Turret/Base***

- Inter-Khepera II message exchange
- Khepera II - host uplink/downlink
- Maximum 115.2 KBPS communication speed
- Collaboration and evolution experiments with multiple robots

Picture 20. IR communication turret/base

A modularized infrared (IR) communication system is used for high-speed inter-robot communication and better management of Khepera experiment. The infrared communication is used for two purposes:

- To communicate between a host computer and multiple Kheperas for management of the experiment
- To exchange signals and messages between multiple Kheperas for inter-robot communication

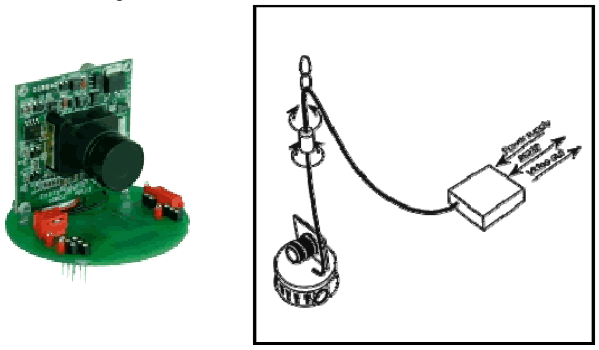
***Radio-modem Turret/Radio Base***

- Radio communication with up to 32 robots
- Direct robot addressing
- Error detection and correction
- Possibility to set a default transmission channel
- Possibility to communicate with a host computer

Picture 21. Radio-modem turret/radio base.

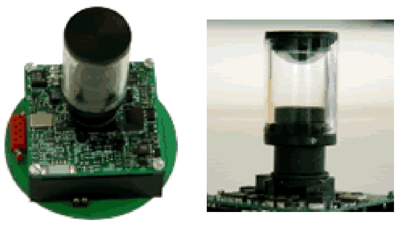
***Video Camera Module***

- 492 x 510 pixels (NTSC)
- Automatic iris control output
- NTSC or PAL format
- Colour or black and white
- Cable with rotating contact for unrestricted motion


Picture 22. Video camera module

***Omni-directional CCD Camera***

In contrast with the conical optic, this cylindrical optic provides a large scale view, from near the robot to the horizon.


Picture 23. Omni-directional CCD camera.

***Wireless Camera Vision Turret***

- Colour vision sensor (380x450 pixels)
- Embedded microphone
- 2.4GHz wireless image transmission to the base receiver
- 2.4GHz wireless audio transmission to the base receiver
- Four transmission channels

***Evolvable Hardware Turret***

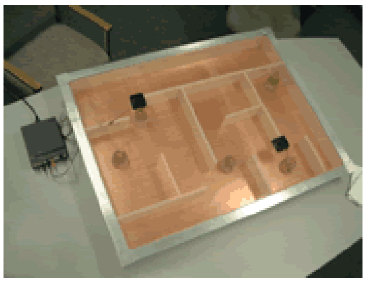
This Khepera add-on turret is used for evolutionary experiments in hardware on a single robot or a group of Khepera robots using FPGAs (Field Programmable Gate Arrays). It handles both "extrinsic" and "intrinsic" evolution of functions, connections, and the physical characteristics of the cells of the FPGA. The results of artificial evolution are tested on Khepera. The Khepera's on-board processor manages the transfer of data to and from the FPGA. All features of the FPGA (cell functions and connections) can be accessed by a user-defined genotype for exploitation by the evolutionary process. The user defines selection and evaluation schemes, as well as the use of a pre-formatted genotype.

- Using Xilinx 6200 series FPGA (Vertex version also available)
- For both "extrinsic" and "intrinsic" hardware evolution
- Full functions of the FPGA are accessible

Picture 24. Evolvable hardware turret.

***Continuous Power Supply***

- Allows several Khepera robots to run simultaneously for a long period of time
- The floor of the maze is a ground (GND), and the VCC is supplied from a copper net that covers the entire maze
- Power is received continuously by each Khepera from a pantograph located on top of the power turret
- The maze is modifiable and adjustable to suit individual experiments
- There is enough experiment space and power for up to 15 Kheperas in a standard size maze

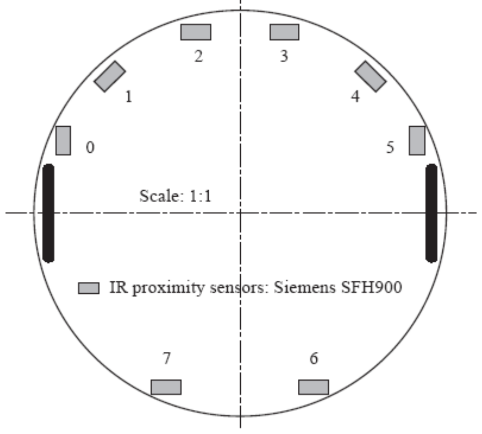

Picture 25. Continuous power supply

***Rotating Contact (cable unroller)***

- Connects Khepera II with a host computer for long running experiments
- Maintains power and communication for 1 or 2 Khepera IIs

d) Typical sensors

Eight Siemens SFH900-2 infra-red proximity sensors are placed around the robot, positioned and numbered as shown in figure below:



Picture 26. Position of 8 IR sensors

e) Accessibility

Very modular at both the software and hardware level, the Khepera has a very efficient library of on-board applications for controlling the robot, monitoring experiments, and downloading new software. A large number of extension modules make it adaptable to a wide range of experimentation.

f) Control methods

To facilitate programming of the robot, several development environments are supported spanning from standard cross-C development to more sophisticated tools like SysQuake, LabVIEW® or MATLAB®, for the 3D WEBOTS Khepera simulator.

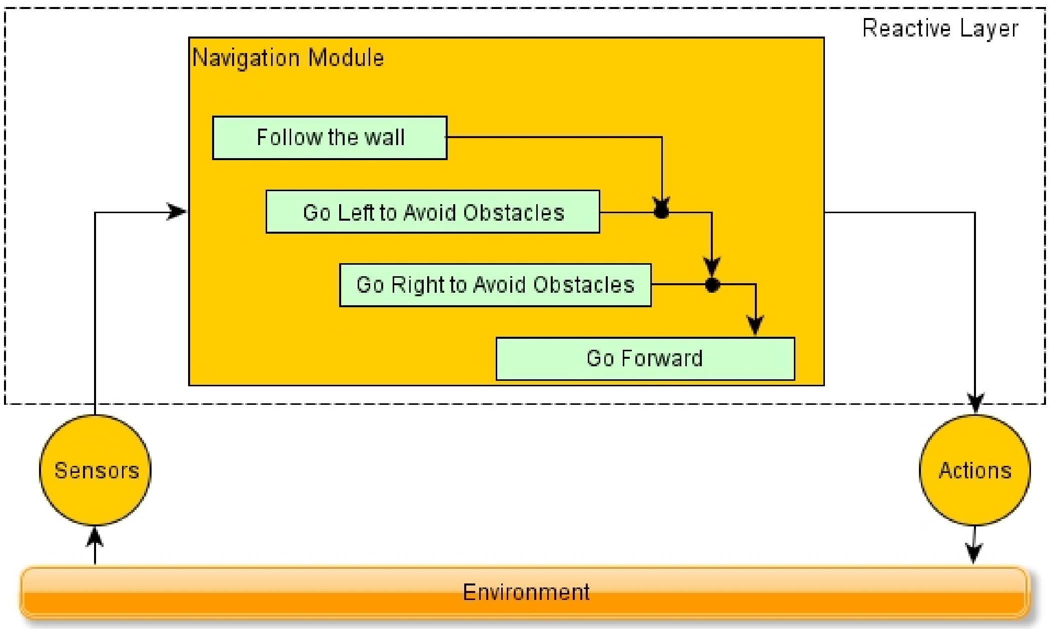
# 5 Fifth Chapter: Empirical Work

The empirical part of this work consists of implementation of the proposed integrated cognitive architecture into a Khepera mobile robot. Due to time limitations, an implementation of only several modules will be described in this paper – the navigation module and the learning mechanism. The learning mechanism will be implemented through the use of genetic programming. I will use a software simulation of the original Khepera robot and its environment. The simulation includes a realistic and tested environment simulation engine and a genetic programming engine that, with few modifications, can be ported to a real Khepera robot.

## 5.1 Implementing the Architecture into a Mobile Robot

### 5.1.1 Navigation

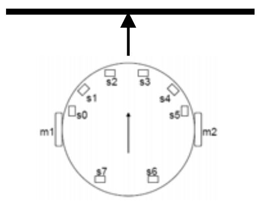
Navigation of the mobile robot is implemented by a subsumption architecture, as presented in the below picture:



Picture 27. Navigation of Khepera robot.

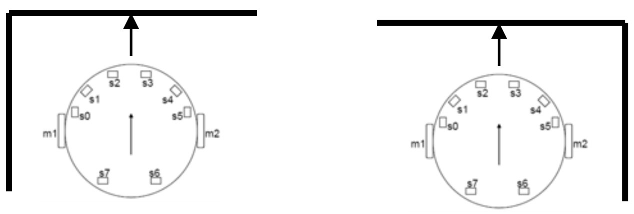
The navigation algorithm is based on several major possibilities, which could happen during the navigation process of a robot:

A. *Obstacle in front of a robot.*



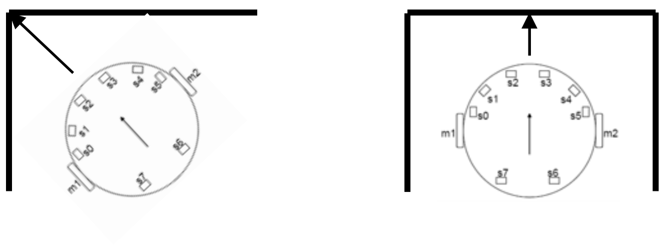
```
IF S[2] < 200 OR S[3] < 200
THEN
        <Turn Right>
        <Go Forward>
ELSE
        <Go Forward>
```

*B. Obstacle in front, robot is in the corner*



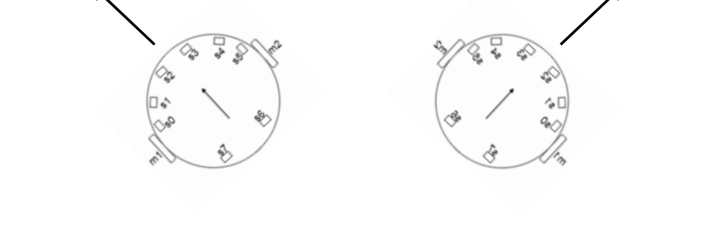
```
IF S[3] < 200 AND (S[2] < 150 OR S[1] < 150 OR S[0] < 150)
THEN
        <Turn Right>
        <Go Forward>

        ELSEIF S[2] < 200 AND (S[3] < 150 OR S[4] < 150 OR S[5] < 150)
        THEN
                <Turn Left>
                <Go Forward>

ELSE
         <Go Forward>
```

*C. Corner in front of a robot* and *D. Deadlock*



These situations are a combination of a situation A and B, thus the earlier mentioned algorithms are convenient for these two situations, too.

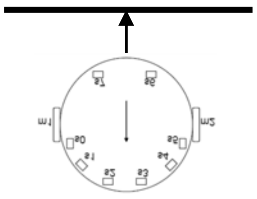
*E. Obstacle on the right/left side of a robot*



```
IF S[4] < 200 OR S[5] < 200
THEN
```

```
          <Turn Left>
          <Go Forward>

          ELSEIF S[0] < 200 OR S[1] < 200
          THEN
                <Turn Right>
                <Go Forward>

  ELSE <Go Forward>
```

F. *Obstacle in back of a robot.*

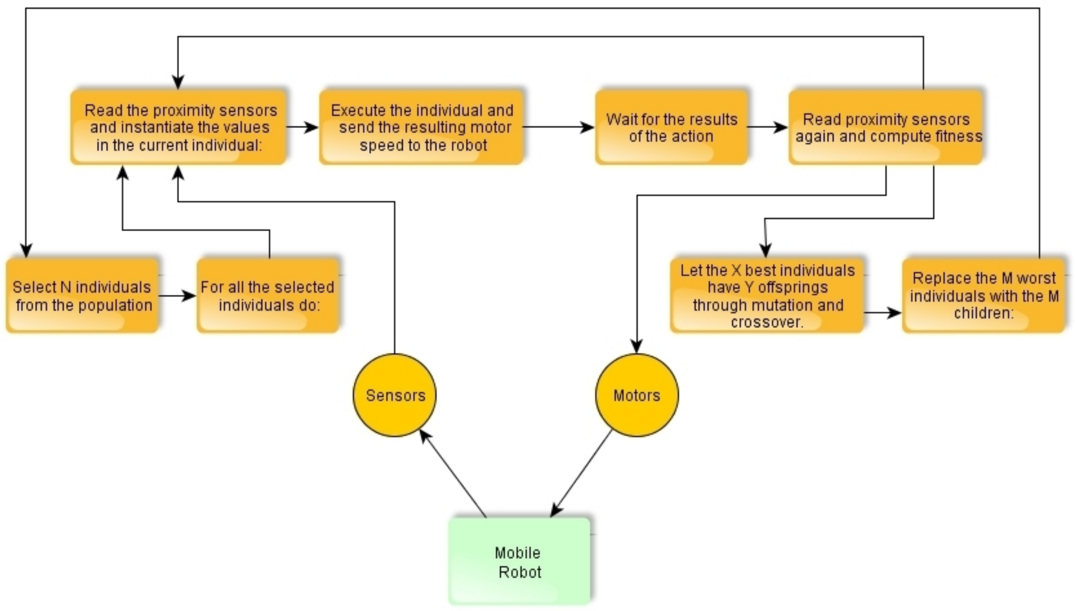


Although this situation is opposite to the situation A, the algorithm for dealing with it is absolutely identical.

The generic algorithm, combining all the described situations, is quite simple and can be presented as follows:

```
IF S[2] < 200 OR S[3] < 200
THEN
        <Turn Right>
        <Go Forward>

        ELSEIF S[3] < 200 AND (S[2] < 150 OR S[1] < 150 OR S[0] < 150)
        THEN
                <Turn Right>
                <Go Forward>

        ELSEIF S[2] < 200 AND (S[3] < 150 OR S[4] < 150 OR S[5] < 150)
        THEN
                <Turn Left>
                <Go Forward>

        ELSEIF S[4] < 200 OR S[5] < 200
        THEN
                 <Turn Left>
                 <Go Forward>

        ELSEIF S[0] < 200 OR S[1] < 200
        THEN
                 <Turn Right>
                 <Go Forward>

ELSE
        <Go Forward>
```

### 5.1.2 Learning through Genetic Programming


Read the proximity sensors and instantiate the values in the current individual:
Execute the individual and send the resulting motor speed to the robot
Wait for the results of the action
Read proximity sensors again and compute fitness
Select N individuals from the population
For all the selected individuals do:
Let the X best individuals have Y offsprings through mutation and crossover.
Replace the M worst individuals with the M children:
Sensors
Motors
Mobile Robot

Picture 28. Genetic learning of Khepera robot.

The goal of GP learning system in the proposed architecture is to evolve the navigational sense-act behaviour into sense-think-act one. GP here tries to approximate the function that takes the vector of sensors values as an input data and returns a vector of motor speeds, controlling the motion of a robot:

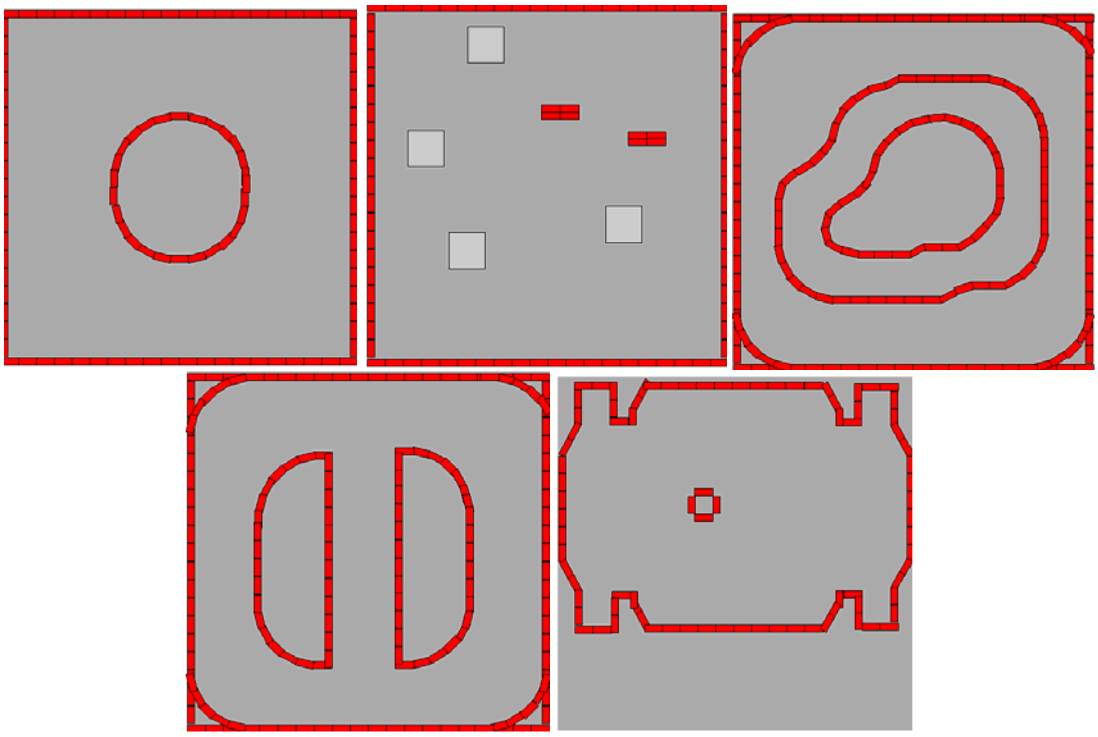
f (S[0], S[1], S[2], S[3], S[4], S[5], S[6], S[7]) = {M[1], M[2]}

The control program code of each individual constitutes the body of the function. The results are compared using a behaviour-based fitness function that measures the accuracy of the approximation by the deviation from desired behaviour of the robot.

## 5.2 Experimental Setup

The experiments were conducted in the simulated environment, using the Khepera GP Simulator for Windows by L.Pilat. The main purpose of the Khepera GP Simulator is to simulate a physical Khepera robot in its environment. The simulation includes sensing of the environment using the robotic sensors and interacting with the environment using the robotic actuators.

The environment of the robot is modelled in a rectangular working area with dimensions 1000x1000mm, on a scale of 0,5. Several worlds are used in the upcoming experiments: a circular wall, constructed of rectangular bricks and used for experimenting with wall following behaviour, and a room with pushable boxes and not-pushable bricks used for experimenting with obstacle avoidance behaviour, the different “maze” rooms used for learning experiments.


Picture 29. World environments, used in experiments.

Two desired chromosome encoding types are used for the upcoming experiments – a tree-based chromosome representation and the method of Module Acquisition (MA).

In the Simulator, trees were defined as recursive node structures where:
1. the root of the tree was a node
2. every node in the tree had zero or more children nodes.

## 5.2.1 Tree-based method

Canonical tree-based GP is a standard method of chromosome representation, described in more details in the part 4.2 of this paper. The trees are created randomly using the full method, which means that all the leaf nodes are of equal length. The height of the trees in the simulator was equal to the maximum creation height and was chosen to be 6, as explained by M.Pilat (2003). M. Pilat (2003) also explains, that “the trees were created by selecting the tree nodes at less than the maximum depth from the function set of parameterized functions. The nodes at maximum depth were chosen from the terminal set. All resulting trees had the greatest allowable height at creation time”.

The list of functions and terminals used in the experiments is provided in the below table:

| **Function Code** | **Description** |
|---|---|
| Add | binary integer addition |
| Sub | binary integer subtraction |
| Mul | binary integer multiplication |
| Div | protected binary integer division |
| AND | binary AND logical operator |

| | |
|---|---|
| OR | binary OR logical operator |
| XOR | binary XOR logical operator |
| << | binary left shift operator |
| >> | binary right shift operator |
| IFLTE | if-less-then-or-equal conditional |
| SetM1 | set motor m1 (only if single output) |
| SetM2 | set motor m2 (only if single output) |

| **Terminal Code** | **Description** |
|---|---|
| Var [s0-s7] | 8 proximity sensors of the robot |
| Var [l0-l7] | 8 ambient light sensors of the robot |
| Const [0-8192] | constant integer value in given range |
| Const [0-100] | constant integer value in given range |

Table 2. Function and terminal codes.

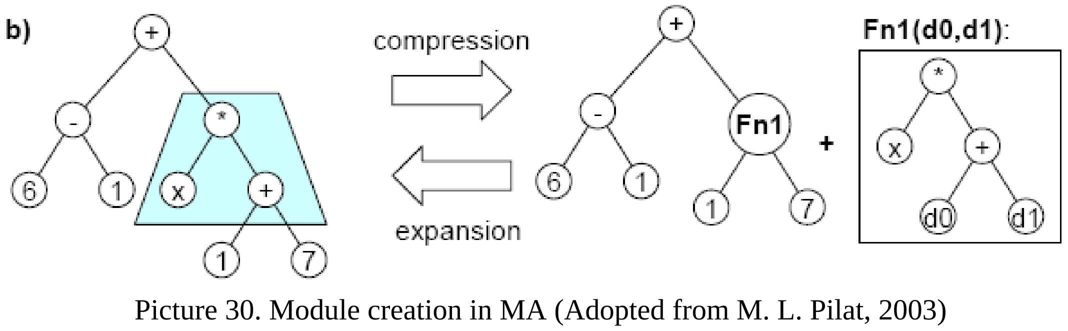
Two genetic operators were used in the tree-based chromosome representation: reproduction and crossover. The crossover operator was a single sub-tree switching crossover between the two parents.

### 5.2.2 Method of Module Acquisition

The Module Acquisition (MA) method was developed by Angeline and Pollack (1992). This method is similar in the simulator to the tree-based method described earlier, as it has the same process of creation of the initial random chromosome, with the same functions and terminals, as explained by M. L. Pilat (2003). However, it differs from the tree-based method by having two additional mutation operators of compression and expansion.

The compression and expansion operators are respectively compressing or expanding a unique module. A module is a sub-tree, which was acquired or selected within an existing individual, defined globally as a unique module and moved to a module library. After moving to the module library modules don't evolve any more. They are kept in the library as long as any individual references them. The number of references expends significantly if the module determines the increase of fitness results in any of its offspring.

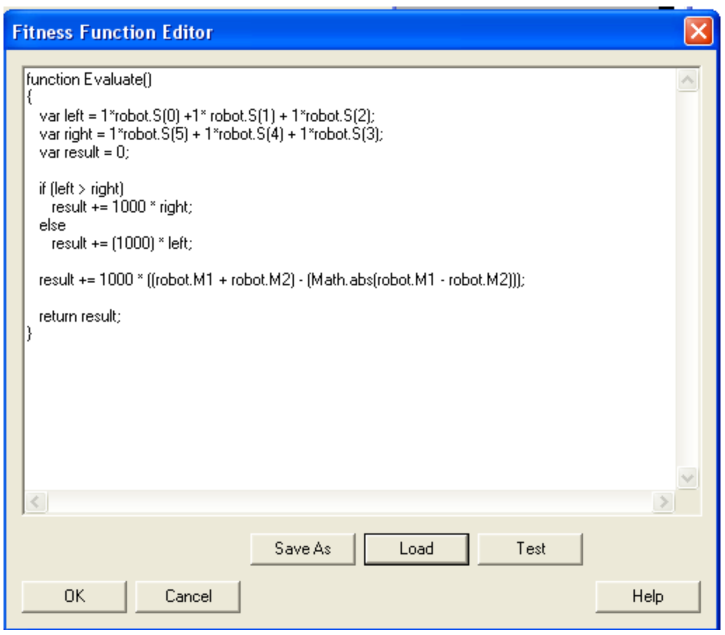
The process of module creation in MA is described in more details in the picture below:



Picture 30. Module creation in MA (Adopted from M. L. Pilat, 2003)

The earlier described navigational algorithm was integrated with the learning behaviour with the use of a fitness function.

The creator of the simulator, M.Pilat provided the possibility to dynamically modify fitness functions at run-time, using a scripting language Microsoft Jscript. This is an object-oriented language,  that supports the ability to create variables and provide a number of pre-defined functions. In the simulator s robot object is made available to the scripting engine to access the sensor and motor values of the robot being studied. An example of a modified fitness function is given in the below picture:



```
function Evaluate()
{
  var left = 1*robot.S(0) +1* robot.S(1) + 1*robot.S(2);
  var right = 1*robot.S(5) + 1*robot.S(4) + 1*robot.S(3);
  var result = 0;

  if (left > right)
    result += 1000 * right;
  else
    result += (1000) * left;

  result += 1000 * ((robot.M1 + robot.M2) - (Math.abs(robot.M1 - robot.M2)));

  return result;
}
```

Picture 31. Fitness Function Editor in the simulator

For experimental setup the crossover percent, mutation percent, tournament size and population size are defined. The properties of a chromosome are also defined. In the experiments several different algorithms of genetic programming are used, such as a linear algorithm and automatically defined functions algorithm. The number of generations used for each experiment is limited to 200.

The following data was collected after each experiment:

- Generation number
- Average and best fitness (if the fitness function was described)
- Number of bumps, i.e. robot's collisions with other objects
- Level of entropy, that measures the state of a dynamic system, represented by the population
- Average and best size of the chromosome
- Average and best structural complexity (SC)
- Average and best evolutional complexity (EC)

## 5.3 Experiment One – Wall Following

In this experiment wall following behaviour is analysed, using two earlier mentioned chromosome representation methods (tree-based and MA) and different fitness functions for each method (no function at all for one task, and different functions for each new task).

For the tree-based method the following parameters were used:

| | |
|---|---|
| **Crossover** | 95% |
| **Mutation** | 2% |
| **Tourn. Size** | 17 |
| **Population** | 2000 |

| | |
|---|---|
| **Chromosome** | |
| **Max. Creation Height** | 6 |
| **Max. Overall Height** | 10 |
| **Number of Outputs** | 6 |
| **Creation Method** | Full |
| **Used functions** | Add,Sub,Mul,Div,IFLTE, AND, OR,XOR,<<, >> |
| **Used terminals** | Var[s0-s7], Const[0-100] |

Table 3. Parameters used for tree-based method.

For the MA method the following parameters were used:

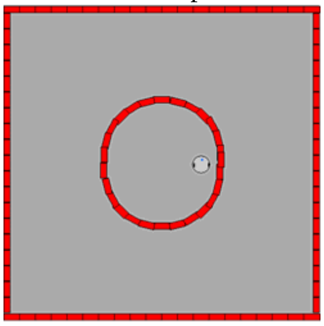
| | |
|---|---|
| **Crossover** | 95% |
| **Mutation** | 2% |
| **Expansion** | 0,05% |
| **Tourn. Size** | 17 |
| **Population** | 2000 |

| | |
|---|---|
| **Chromosome** | |
| **Max. Creation Height** | 6 |
| **Max. Overall Height** | 10 |
| **Number of Outputs** | 6 |
| **Creation Method** | Full |
| **Used functions** | Add,Sub,Mul,Div,IFLTE, AND, OR,XOR,<<, >> |
| **Used terminals** | Var[s0-s7], Const[0-100] |

Table 4. Parameters used for model acquisition method.

Three different fitness functions were used for each of the two methods (the source codes of the fitness functions used are presented in Appendix A1) and one experiment was performed with each method without any fitness function. The length of each experiment was restricted to 55 generations.

The world environment used for the experiment is shown in the below picture:



Picture 32. Environment used for wall following experiment.

The robot’s start position for each new simulation was defined as: X=625 (mm-world), Y=505 (mm-world), Alpha=90 (degree).

## 5.4 Experiment Two – Learning

For evaluating the learning ability of the robot and compare this ability between the robots in different learning situations, the following experiment was performed: the robot was learned to navigate in three different environments for 25 generations and after each learning it was placed in the "maze" environment and its navigation abilities were observed for 25 generations maximum, or until the way through the maze was found by the robot if it happened earlier.

For the tree-based method of genetic learning the following parameters were used:

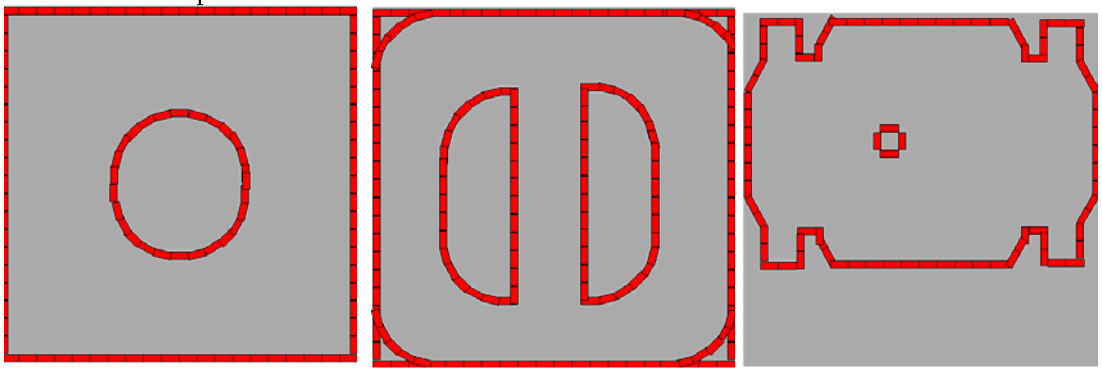
| | |
|---|---|
| **Crossover** | 95% |
| **Mutation** | 2% |
| **Tourn. Size** | 17 |
| **Population** | 2000 |

| **Chromosome** | |
|---|---|
| **Max. Creation Height** | 6 |
| **Max. Overall Height** | 10 |
| **Number of Outputs** | 6 |
| **Creation Method** | Full |
| **Used functions** | Add,Sub,Mul,Div,IFLTE, AND, OR,XOR,<<, >> |
| **Used terminals** | Var[s0-s7], Const[0-100] |

Table 5. Parameters used for learning experiment.

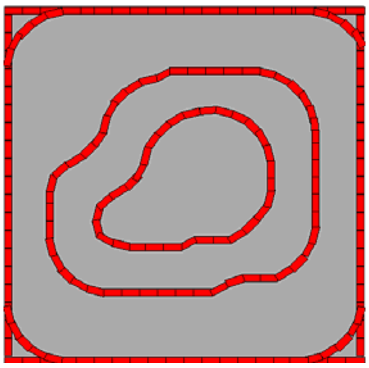
Three different fitness functions were used for each of the two methods (the source code of the fitness function used is presented in Appendix A1 as fitness function 2) and one experiment was performed with each method without any fitness function. The length of each experiment was restricted to 50 generations. If robot could fulfil the task of navigating through the maze earlier, the experiment was finished earlier.

Three world environment used for learning the navigation behaviour are shown in the below picture:



Picture 33. Environments used for training of navigation behaviour.

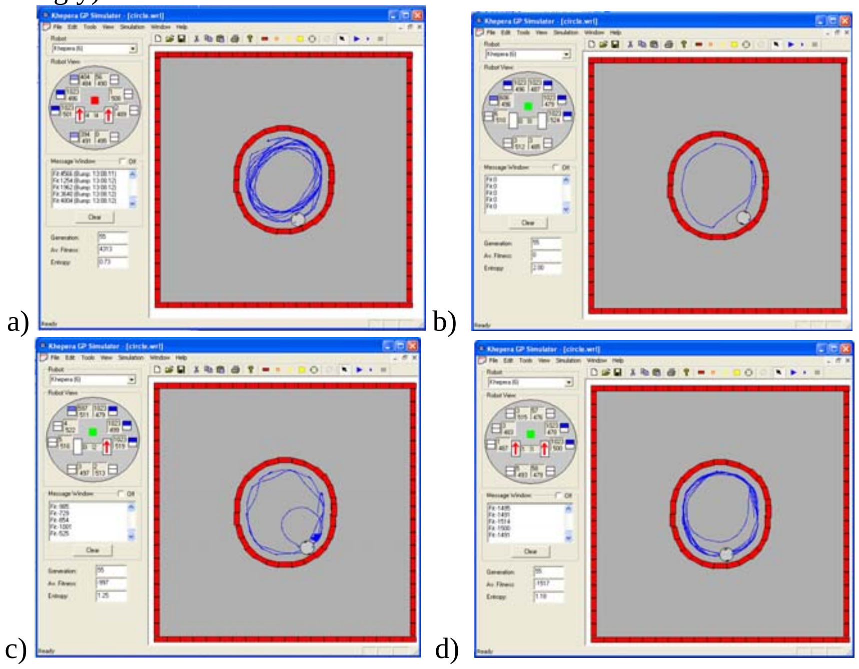
The "maze" environment, used for evaluating the earlier learning, is shown below:


Picture 34. Environments used for evaluating the learned behaviour.

# 6 SIXTH CHAPTER: RESULTS AND DISCUSSION

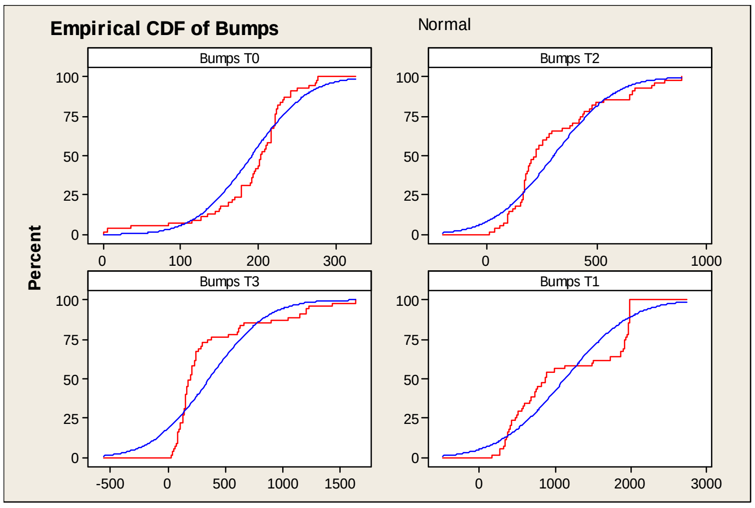
## 6.1 Wall Following Experiments

The pictures below present the screenshots of a tree-based method's simulations with fitness functions T0, T1, T2 AND T3 (here letter T corresponds to tree-based method, and numbers 0,1,2 and 3 correspond to fitness functions No. 0,1,2 and 3 accordingly):



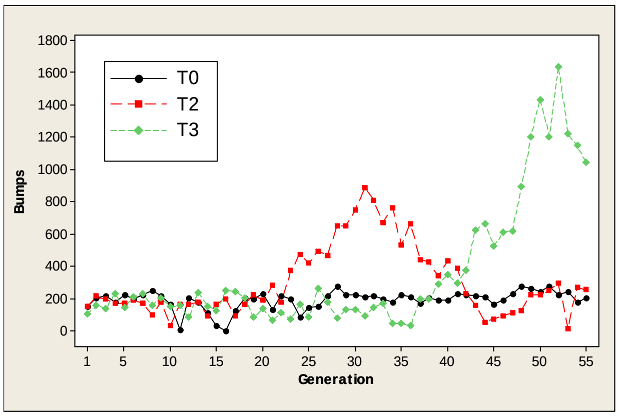
Picture 35. Wall following experiment result for tree-based method. a-T0, b-T1, c-T2, d-T3

The biggest number of bumps was observed during the simulation with a fitness function T1. However, an empirical CDF showed that only the data of bumps for functions T0, T2 and T3 can be described as normally distributed:



Picture 36. Empirical CDF of bumps, tree-based method.

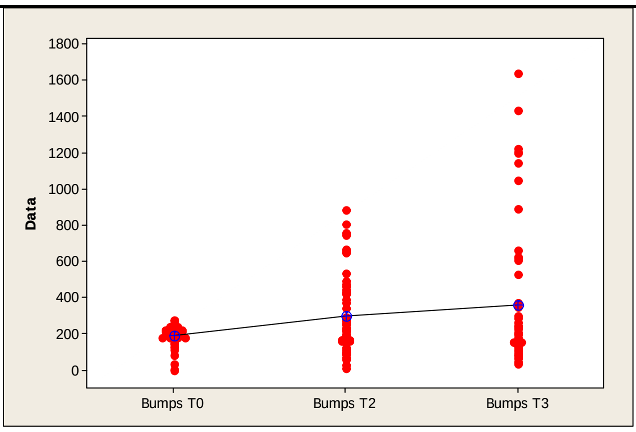
From that sample, the biggest amount of bumps during the experiments was received by the robot with fitness function T3. The robot without any fitness function defined had a stable number of bumps all over the period of the experiment. The number of bumps for the robot with T2 fitness function has increased for a short period between generations 25 and 41, and later decreased back to the level of T0:



Picture 37. Number of bumps, tree-based method.

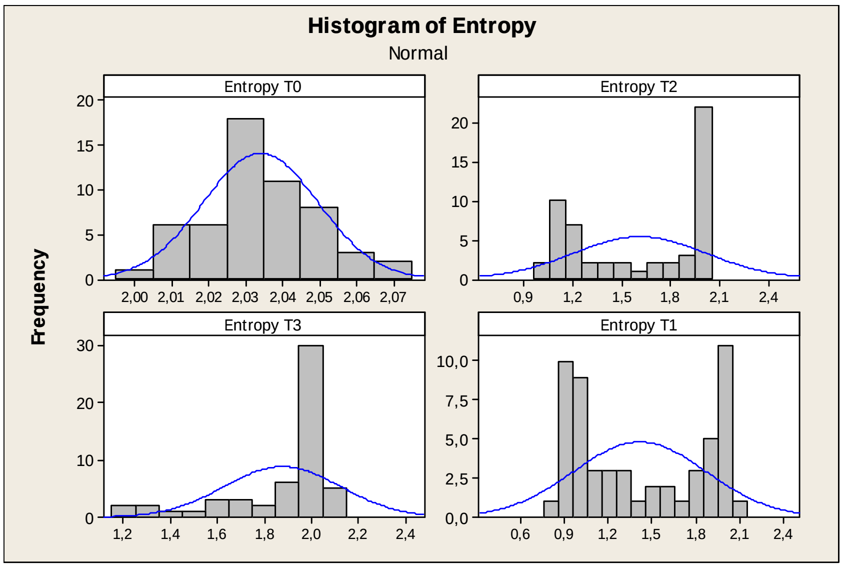
The one-way Anova test was performed for analysing whether the different fitness functions are affecting the robot's navigating ability, i.e. number of bumps during navigating. The result of a very low p-value between the groups proves that there is a statistically significant difference between the data of different fitness functions. That means that the independent variable of fitness function does affect the response variable of a number of bumps:

| *Source of Variation* | *SS* | *df* | *MS* | *F* | *P-value* | *F crit* |
|---|---|---|---|---|---|---|
| Between Groups | 802235 | 2 | 401117,4788 | 5,78 | **0,004** | 3,05 |
| Within Groups | 11245530 | 162 | 69416,85 | | | |
| | | | | | | |
| Total | 12047765 | 164 | | | | |



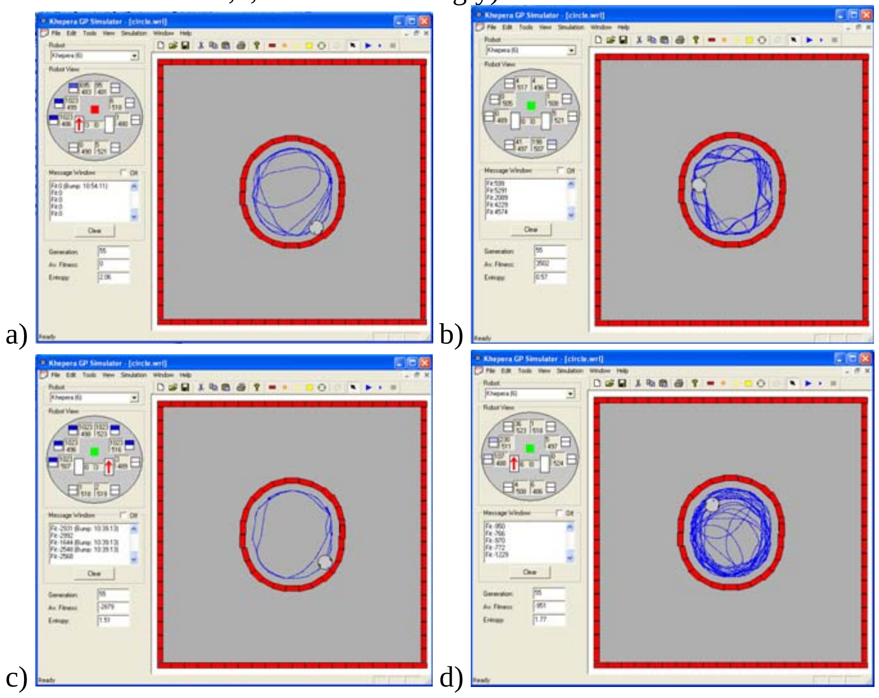
Picture 38. One-way Anova test results of wall following experiment, tree-based method.

The analysis of entropy level shows that the distribution of data becomes abnormal as the fitness function is used for improving the robot's navigation. The histogram of entropy's data proves this, as shown in the picture below:



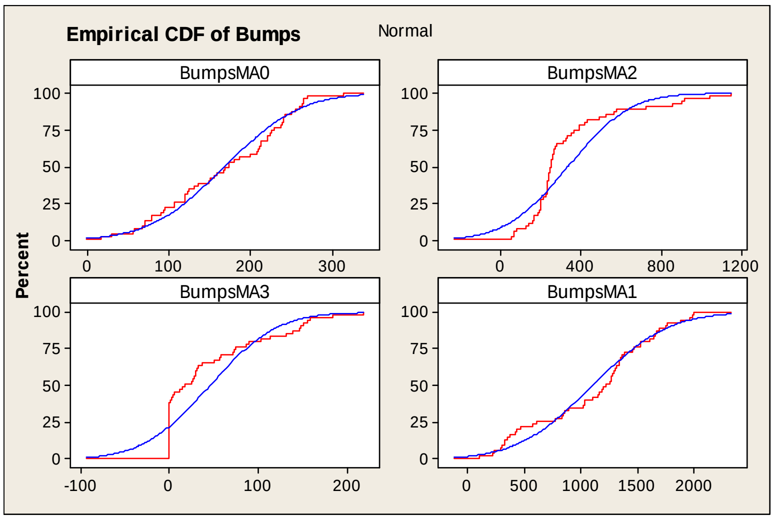
Picture 39. Empirical CDF of entropy data, tree-based method.

The pictures below present the screenshots of MA method's simulations with fitness functions MA0, MA1, MA2 AND MA3 accordingly (here letters MA correspond to Module Acquisition method, and numbers 0,1,2 and 3 correspond to fitness functions No. 0,1,2 and 3 accordingly):



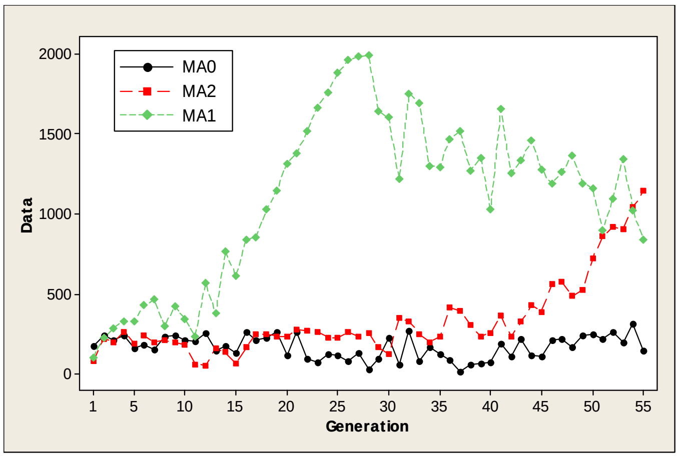
Picture 40. Wall following experiment result for MA method. a-MA0, b-MA1, c-MA2, d-MA3

It is possible to say that the data of the number of bumps for the MA method is distributed normally for the robots with fitness functions MA0, MA1 and MA2. The CDF in the below picture presents the graphs for the data from all the four wall following experiments:



Picture 41. Empirical CDF of bumps, MA method.

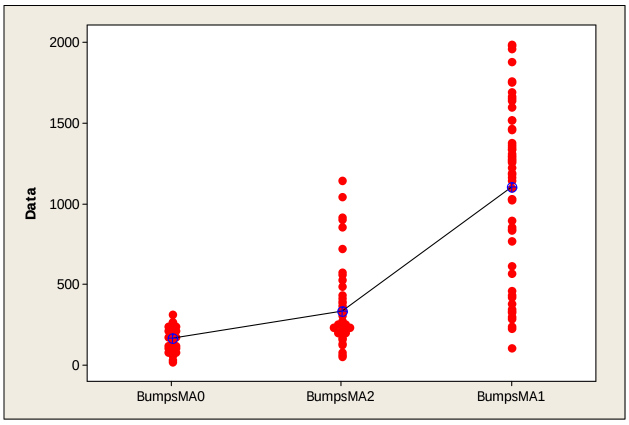
Out of these three data sets the number of bumps of MA1 is the biggest one, the robot with MA2 fitness function had significantly less bumps during its way, although beginning from the generation 45 the number of bumps increased and reached the level of MA1:



Picture 42. Number of bumps, MA method.

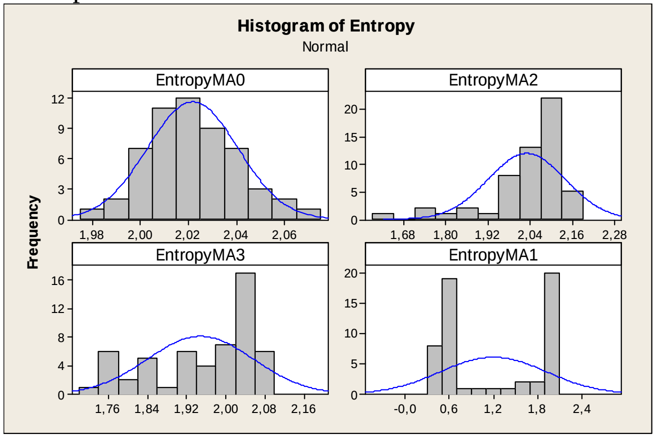
The one-way Anova test was performed again for analysing whether the different fitness functions are affecting the number of bumps during navigating, when using MA method. The very low p-value result proved again that there is a statistically significant difference between the data of different fitness functions. That means that the independent variable of fitness function does affect the response variable of a number of bumps:

| *Source of Variation* | *SS* | *df* | *MS* | *F* | *P-value* | *F crit* |
|---|---|---|---|---|---|---|
| Between Groups | 2307611 | 2 | 1153806 | 50,93 | **6,92E-18** | 3,05 |
| Within Groups | 3670209 | 162 | 22655,61 | | | |
| | | | | | | |
| Total | 5977820 | 164 | | | | |



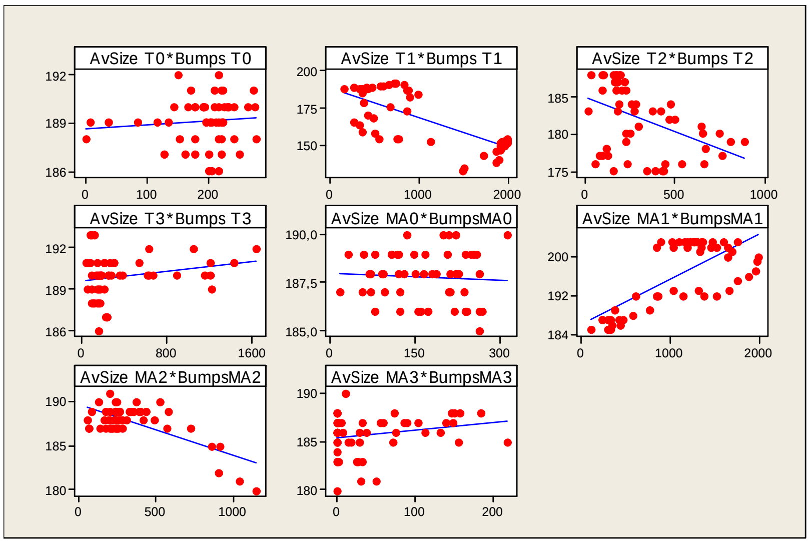
Picture 43. One-way Anova test results of wall following experiment, MA method

The analysis of entropy level shows the same result as for a tree-based method: the distribution of data becomes abnormal as the fitness function is used for improving the robot's navigation. The picture of entropy data histograms compared to a normal distribution line is presented below:



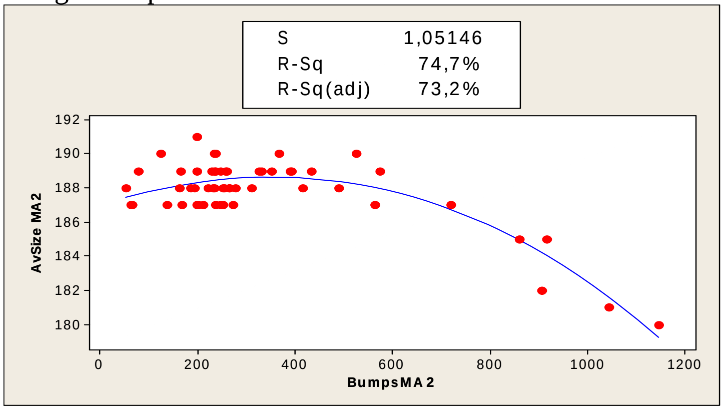
Picture 44. Histogram of entropy, MA method.

The results of a regression test show that there is a correlation between the number of bumps and the average size of the generation for the robots with fitness functions 1 and 2, with both tree-based and MA method. In the picture below the regression lines in these cases are steeper, and the steeper is the line, the stronger is the relation:



Picture 45. Regression test results

For example, the R-square value for the robot with fitness function MA2 is 73,2%. It means that the number of bumps explains the 73,2% of the average size data, received during the experiments:



Picture 46. Regression test results, MA2 fitness function.

The two-way Anova test was performed in order to analyze if there is an correlation between the number of bumps and both the different fitness functions and different methods. The results of the two-way Anova variance table are presented below:

**Two-way ANOVA: Bumps versus Method; Fitness Fnc**

```
Source        DF         SS        MS       F      P
Method         1     746544    746544    5,79  0,017
Fitness Fnc    3   65138735  21712912  168,34  0,000
Interaction    3    2021638    673879    5,22  0,001
Error        432   55721301    128984
Total        439  123628219

S = 359,1   R-Sq = 54,93%   R-Sq(adj) = 54,20%
```

Table 6. Two-way Anova test result for bumps.

These results show that all three p-values are less than commonly used value of 0,05, which mean that there is a significant evidence for both method's and fitness function's influence on the number of bumps. There is also significant evidence that the interaction of these two variables also has the influence on the number of bumps.

The two-way Anova test was performed once more for analyzing the influence of method and fitness function on the best size of generation. The results are presented in the below variance table:

**Two-way ANOVA: Best Size versus Method; Fitness Fnc**

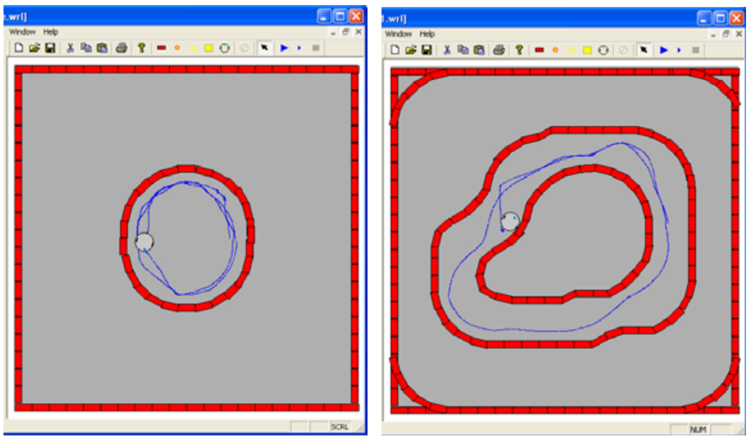
```
Source        DF      SS       MS     F      P
Method         1   10437  10437,4  8,54  0,004
Fitness Fnc    3    8638   2879,4  2,36  0,071
Interaction    3   19699   6566,3  5,37  0,001
Error        432  527795   1221,7
Total        439  566569

S = 34,95   R-Sq = 6,84%   R-Sq(adj) = 5,33%
```

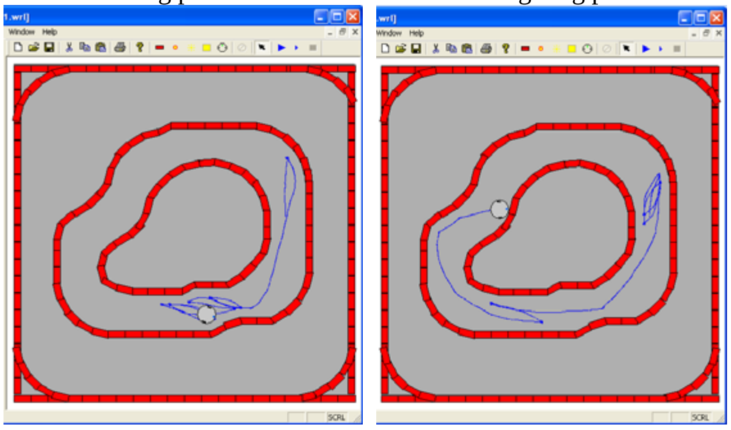
Table 7. Two-way Anova test result for best size.

The results show that there is no significant evidence of fitness function's influence on the best size of generation. However, the method is influencing the best size, the same as the interaction between method and fitness function.

## 6.2 Learning Experiments

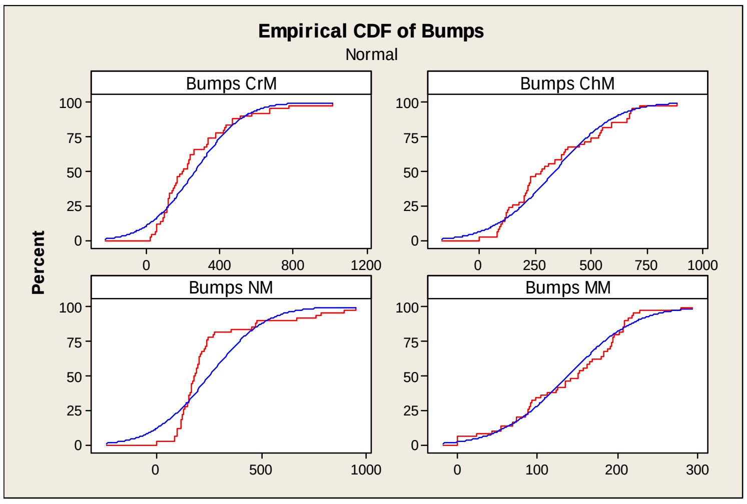
The pictures below present the screenshots of most interesting simulated experiments of robot's training in two different environments and the results of their learning in the "maze" environment. The screenshots of other two experiments can be found in Appendix A2.

a)

training phase navigating phase

b)

training phase navigating phase

Picture 47. Screenshots of learning experiments.

The data of the number of bumps during the experiments can be considered as normally distributed (here and later label CrM means "circle" environment, ChM – "chalways" environment, NM – "Nordin" environment and MM – "maze" environment):

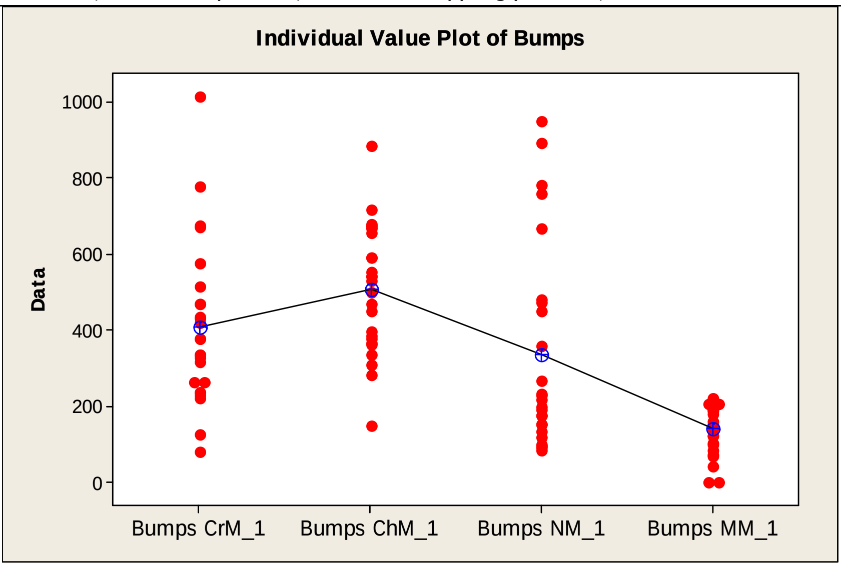


Picture 48. Empirical CDF of bumps, learning experiment.

The one-way Anova test shows that the type of training influences the number of bumps, occurring during navigation after it:

**One-way ANOVA:**

```
Source  DF       SS      MS      F      P
Factor   3  1802355  600785  15,96  0,000
Error   96  3614855   37655
Total   99  5417210

S = 194,0   R-Sq = 33,27%   R-Sq(adj) = 31,19%
```



Picture 49. One-way Anova test results of wall following experiment for number of bumps.

There is also a slight influence of the training on the best fitness value:

**One-way ANOVA:**

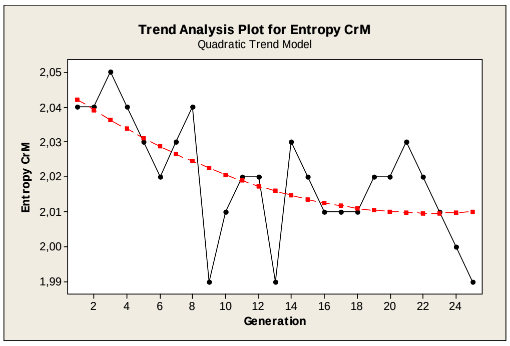
```
Source  DF         SS        MS     F      P
Factor   3   14815179   4938393  2,82  0,043
Error   96  167968189   1749669
Total   99  182783367

S = 1323   R-Sq = 8,11%   R-Sq(adj) = 5,23%
```

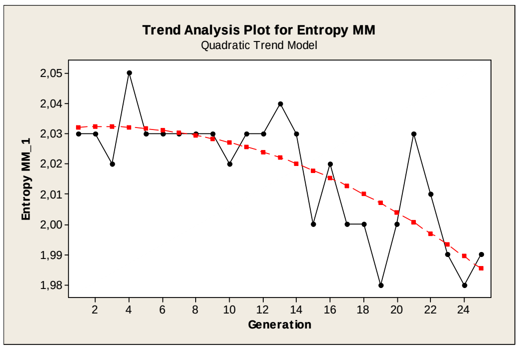
Table 8. One-way Anova test result for best fitness value.

From the screenshots (picture 48) it's possible to see that training in a circle environment was more effective for future navigating in the "maze", than training in the "maze" itself. After training in the small circle environment, the robot was able to go through the "maze" without any significant troubles during the period of 12 generations. After training in the "maze", the robot was not able to find the way through it during 25 generations and lost his way between the walls for two times during this period.

Although the visual results of a circle-training effectiveness were self-evident, the statistical data of the experiments was analysed in more details.

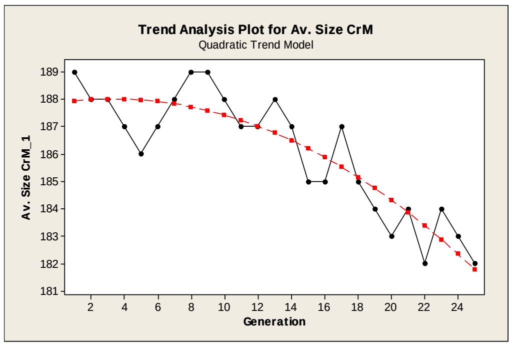
Trend analysis highlighted the major differences between the circle-training and maze-training situations for entropy, bumps and average size of generation. The quadratic trend line shows that the entropy level of the maze-training case is decreasing fast, while the entropy of a circle-training system has a tendency to stabilize:

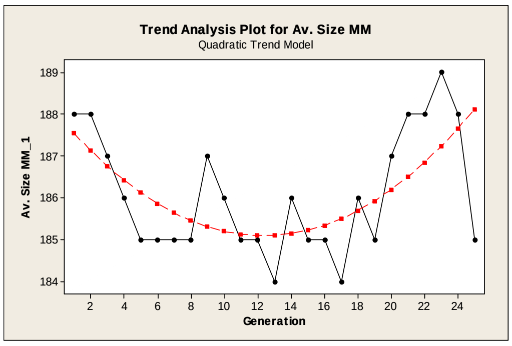




Picture 50. Trend analysis for entropy level.

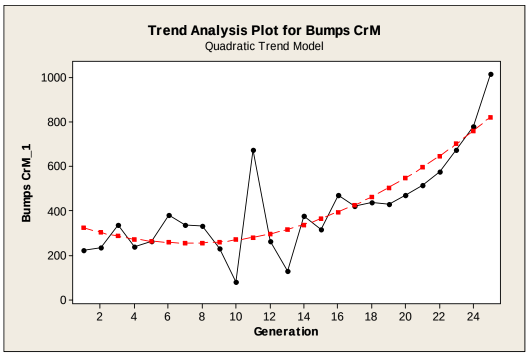
The quadratic trend analysis of an average size of a generation shows that for a circle-training system the average size of generation has an decreasing trend, while for a maze-training one the there is a tendency to increase:

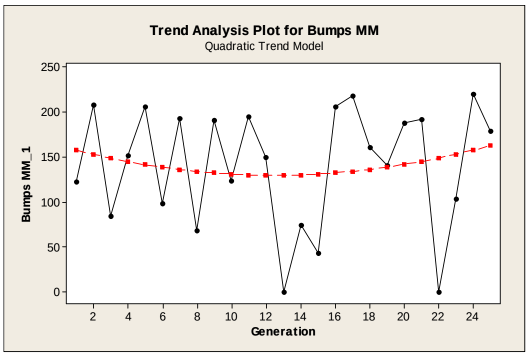




Picture 51. Trend analysis for average size.

The trend analysis of the number of bumps for both cases showed differences between them, too. While the number of bumps in a maze-training case has a tendency to remain stable, the trend of a number of bumps in a circle-training case is increasing:





Picture 52. Trend analysis for number of bumps.

To summarize, the trend analysis shows that the maze-training model has a trend to perform more efficiently than the circle-training model, with less bumps and higher stability level. However, the practical results of the experiments show the opposite results, as in the circle-training model the robot is performing better.

The possible explanation for such results could be like this: in order to get fast and quite good results in unknown environment, it is useful to train the robot in a small area. Thus the robot learns all the principal navigation rules and can better solve simple navigation problems in future. The visual results of earlier experiment prove this. However, such training is not good enough in a long-term. It’s possible to suppose, that if it is necessary to learn the robot to navigate efficiently for a long period of time in future, it is better to train him for a longer period of time, and the training should be performed in the environment, which is as close as possible to the future environment. However, further experiments should be performed in order to prove this hypothesis.

# 7 SEVENTH CHAPTER: EPILOGUE

## 7.1 Conclusions

The results of successful implementation of several modules of the proposed integrated cognitive architecture into a mobile robot indicated the plausibility of the architecture.

For robot control, the results indicated that different genetic programming methods can be successfully integrated with a subsumption architecture and used as a learning mechanism. There is a statistically significant difference in navigational performance when using different genetic programming methods, so different methods could be used for achieving different goals.

Different fitness functions, from the other side, can be used for designing desirable navigational behaviour, such as wall following or obstacle avoidance. Both genetic programming methods and fitness functions are influencing the quality of robot navigation behaviour. The interaction of these two factors also had a significant influence on the navigation performance.

The results of learning experiments highlighted the influence of specific training environments on the future navigation performance. Although it seemed natural before the experiment that best training results should be reached in the same environment as the one used for the future task, but the experimental results did not prove such a hypothesis. The results of the learning experiment enable to suggest that training in a small area helps to obtain better short-term navigational performance, while a longer training in the same environment as the one used for the future task helps to obtain better long-term navigational performance.

## 7.2 Future work

Improvement of navigation ability of the robot can be further investigated, with the experiments in a real-world environment. The results of the empirical work show that there is a connection between fitness functions and the navigational performance. Finding the optimal fitness function for a desirable navigational behaviour could be a possible direction for the future work.

There also can be many future directions of the conducted research keeping in mind that the navigation ability is the only one possible cognitive ability of the mobile robot that can be implemented with a help of integrated cognitive architecture. The important direction for the future work is developing and implementing the rest of the proposed architecture and evaluating the architecture's performance.

One more possible direction for the future work is integrating different learning mechanisms, such as supervised learning, unsupervised learning, or reinforcement learning.

The further investigating of emotions influence on different cognitive abilities and their role in forming self-awareness is defined as one more direction for the future work.

# REFERENCES

1. Ah-Hwee Tan, Gee Wah Ng., 2010. *A Biologically-Inspired Cognitive Agent Model Integrating Declarative Knowledge and Reinforcement Learning*. In Proceedings of IAT'2010. pp.248~251
2. Amir, E. and Maynard-Zhang, P., 2004. *Logic-Based Subsumption Architecture*. Artificial Intelligence (AIJ), 153(1-2), pp. 167-237.
3. Anderson, J., 2003. *Embodied cognition: a field guide*. Artif Intell 149, pp.91–130.
4. Anderson, J. et al., 2004. *An integrated theory of the mind*. Psychol Rev 111, pp. 1036–1060
5. Andreae, J, 1998. *Associative Learning for a Robot Intelligence*. Imperial College Press
6. Angeline, P. and Pollack, J., 1992. *The evolutionary induction of subroutines*. In Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society, Bloomington, Indiana, USA. Lawrence Erlbaum.
7. Arena, P., 2008. *Dynamical Systems, Wave-Based Computation and Neuro-Inspired Robot,* SpringerWien NewYork.
8. Bach, J., 2009. *Principles of Synthetic Intelligence PSI: An Architecture of Motivated Cognition*. Oxford University Press, Inc.
9. Belavkin, R., Ritter, F., 2000. *Adding a theory of motivation to ACT-R*. In Proceedings of the Seventh Annual ACT-R Workshop. 133–139. Department of Psychology, Carnegie-Mellon University: Pittsburgh, PA
10. Brachman, R., 2002. *Systems that know what they're doing*, In IEEE Intelligent Systems, vol. 17, no. 6, pp. 67–71.
11. Bratman, M., Israel, D., Pollack J., 1988. *Plans and resource-bounded practical reasoning*. Comput Intell 4(4), pp. 349–355
12. Brooks, R., 1999. *Cambrian intelligence: the early history of the new AI*. MIT Press, Boston, MA.
13. Buiu, C., 2009. *Towards Integrated Biologically Inspired Cognitive Architectures.* ECAI 2009 – International Conference – 3rd Edition.
14. Butler, G., Gantchev, A., Grogona, P., 2001. *Object-oriented design of the subsumption architecture*. Softw Pract Exper 31, pp. 911–923
15. Duch, W. et al., 2008. *Cognitive Architectures: Where do we go from here?* Proceeding of the 2008 conference on Artificial General Intelligence, 2008.
16. Dudek and Jenkin, 2000. *Computational Principles of Mobile Robotics.*Cambridge University Press.
17. Edeman, G., 1993. *Neural Darwinism: Selection and re-entrant signalling inhigher brain function,* Neuron, Vol. 10, pp. 115-125.
18. Franceschini, R., McBride, D., Sheldon, E., 2001. *Recreating the Vincennes Incident using affective computer generated forces*. In Proceedings of the 10th Computer Generated Forces and Behavioral Representation Conference. 10TH-CG-044.doc. Orlando, FL: Division of Continuing Education, University of Central Florida.
19. Friedlander, D., Franklin, S., in: Ben Goertzel, Pei Wang (Eds.), 2008. *LIDA and a theory of mind,* in: Artificial General Intelligence (AGI-08), IOS Press, Memphis, TN, USA.
20. Goertzel, B. et al., 2010. *World survey of artificial brains, Part II: Biologically inspired cognitive architectures,* Neurocomputing.
21. Gratch, J., Marsella, S., 2001. *Modeling emotions in the Mission Rehearsal Exercise*. In Proceedings of the 10th Computer Generated Forces and Behavioral Representation Conference (10TH—CG-057). Orlando, FL: University of Central Florida, Division of Continuing Education.
22. Hollnagel, E., Woods, D., 1999. *Cognitive Systems Engineering: New wine in new bottles*. Int. J. Hum.-Comput. Stud., pp. 339~356

23. Hui-Qing Chong et al., 2009. *Integrated cognitive architectures: a survey*, Springer Science+Business Media B.V
24. Jilk, D. et al., 2008. *SAL: an explicitly pluralistic cognitive architecture*, Journal of Experimental & Theoretical Artificial Intelligence, Volume 20, Issue 3, pp. 197 – 218.
25. Jones, R., 1998. *Modeling pilot fatigue with a synthetic behavior model*. In Proceedings of the 7th Conference on Computer Generated Forces and Behavioral Representation, pp. 349–357. Orlando, FL: University of Central Florida, Division of Continuing Education
26. Kamat, R., 2010. *Modelling and Analyzing of Biologically Inspired Robot*, International Journal of Advanced Computer and Mathematical Sciences, Vol 1, Issue 1, Dec, 2010, pp 7-11
27. Koza, J., 1992. *Genetic Programming: On the programming of computers by means of natural selection*, MIT Press.
28. Koza, J., 1994. *Genetic Programming II*, MIT Press.
29. K-Team, 1999. *Khepera User Manual*, Version 5.02.
30. Laird, J., Rosenbloom, P., Newell, A., 1986a. *Chunking in Soar: The anatomy of a general learning mechanism*. Mach Learn 1, p.11–46
31. Laird, J., Rosenbloom, P., Newell, A., 1986b. *Universal subgoaling and chunking: the automatic generation and learning of goal hierarchies*. Kluwer, Boston, MA
32. Laird, J., 2006. *BICA: Biologically-inspired Cognitive Architecture*. 26th SOAR Workshop, 2006.
33. Langley, P. and Messina, E., 2004. *Experimental Studies of Integrated Cognitive Systems*. Proceedings of the Performance Metrics for Intelligent Systems Workshop
34. Langley P, Choi D, 2006. *A unified cognitive architecture for physical agents*. In: Proceedings, twenty-first national conference on artificial intelligence, pp 1469–1474
35. Lehman, J, Laird, J, Rosenbloom P., 2006. *A gentle introduction to Soar, an architcture for human cognition: 2006 update*. Retrieved 17 May 2011 from http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/GentleIntroduction-2006.pdf
36. Mitchell, M., 1996. *An introduction to genetic algorithms*, New York MIT Press.
37. Mondada, F., Franzi, E., 1993. *Biologically Inspired Mobile Robot Control Algorithms*, in NFP-PNR 23 Symposium
38. Nehmzow, U., 2006. *Scientific methods in mobile robotics: quantitative analysis of agent behaviour*, Springer-Verlag London Limited.
39. Newell, A., 1973. *Production Systems: Models of Control Structures*. In W. G. Chase (ed.): Visual Information Processing, New York: Academic Press, pp. 463–526
40. Newell, A., 1987. *Unified Theories of Cognition*, Cambridge, MA: Harvard University Press.
41. Newell, A., Simon, H., 1976. *Computer Science as Empirical Inquiry: Symbols and Search*, Communications of the ACM, vol. 19 (March 1976), pp. 113—126
42. Nordin, P., Banzhaf, W., 1997. *Real Time Control of a Khepera Robot using Genetic Programming*. Cybernetics and Control, Vol. 26, Dortmund, Germny.
43. Norling, E., Ritter, F., 2004. *Towards Supporting Psychologically Plausible Variability in Agent-Based Human Modelling*. AAMAS 2004, pp. 758–765
44. Orazio Migl et al., 1995. *Evolving Mobile Robots in Simulated and Real Environments*, Artificial Life, Volume 2 Issue 4, Summer 1995
45. Pilat, M., L., 2003. *Hierarchical Learning Systems*. Carleton University, Ottawa, Ontario.
46. Rao, A., Georgeff, M., 1991. *Modeling rational agents within a bdi-architecture*. In: Proceedings, second international conference on principles of knowledge representation and reasoning. Morgan Kaufmann, San Mateo, CA, pp 473–484
47. Rolfe, R., and Haugh, B., 2003. *Integrated Cognition: Proposed Definition of Ingredients*. Draft document, Institute for Defense Analyses, June 5, 2003.

48. Rosenbloom, P., 1998. *Emotion in Soar*. In Proceedings of Soar Workshop 18, 26–28. Vienna, VA: Explore Reasoning Systems
49. Roth, E., Patterson, E., and Mumaw, R., 2001. *Cognitive Engineering: Issues in User-Centered System Design*. In J.J. Marciniak, Ed. Encyclopedia of Software Engineering (2nd Edition). NY: John Wiley and Sons.
50. Rumelhart, D., McClelland, J. and the PDP Research Group, 1986. *Parallel Distributed Processing*, (Vols. 1&2), Cambridge, Massachusetts: MIT Press.
51. Sun, R., Peterson, T., 1996. *Learning in reactive sequential decision tasks: the CLARION model*. In: Proceedings, IEEE international conference on neural networks, pp 1073–1078
52. Sun, R., Peterson, T., 1998. *Hybrid learning incorporating neural and symbolic processes*. In: Proceedings, IEEE international conference on fuzzy systems, pp. 727–732
53. Sun, R., Merrill, E., Peterson, T., 2001. *From implicit skills to explicit knowledge: a bottom-up model of skill learning*. Cogn Sci 25(2), pp. 203–244
54. Sun, R., 2003. *A tutorial on CLARION 5.0*. Tech. rep., Cognitive Science Department, Rensselaer Polytechnic Institute
55. Sun, R., Slusarz, P., Terry, C., 2005. *The interaction of the explicit and the implicit in skill learning: a dual-process approach*. Psychol Rev 112(1), pp.159–192
56. Sun, R., 2006. *Cognition and Multi-Agent Interaction: From Cognitive Modelling to Social Simulation*, Cambridge University Press, New York.
57. Sun, R., Zhang, X., 2006. *Accounting for a variety of reasoning data within a cognitive architecture*. J Exp Theor Artif Intell 18(2), pp.169–191
58. Sun, R., 2007. *The Challenges of Building Computational Cognitive Architectures*, in Studies in Computational Intelligence, Volume 63, 37 -60.
59. Vernon, D. et al., 2007. *A Survey of Artificial Cognitive Systems: Implications for the Autonomous Development of Mental Capabilities in Computational Agents*. In IEEE Trans. Evolutionary Computation 2007:11(2), pp. 151-180.
60. Wayne, D. Gray, 2007. *Integrated Models of Cognitive Systems*, OUP.
61. Wilson, N. Tutorial: The CLARION Cognitive Architecture
62. http://cordis.europa.eu/fp7/ict/cognition/
63. http://en.wikipedia.org/wiki/Cognition
64. http://sitemaker.umich.edu/soar/home
65. http://en.wikipedia.org/wiki/LIDA_(cognitive_architecture)
66. http://www.mobilerobots.com

# APPENDIX A1

***Fitness functions used in experiments.***

1. *Fitness function 1:*

```
function Evaluate()
{
  var left = 1*robot.S(0) +1* robot.S(1) + 1*robot.S(2);
  var right = 1*robot.S(5) + 1*robot.S(4) + 1*robot.S(3);
  var result = 0;

  if (right > 1023)
    result += 1000 - right;
  else if (right < 20)
    result += (1000/20) * right;
  else
    result += 1000;

  if (left > 1023)
    result += 1000 - left;
  else if (left < 20)
    result += (1000/20) * left;
  else
    result += 1000;

  result += 1000 * ((robot.M1 + robot.M2) - (Math.abs(robot.M1 - robot.M2)));

  return result;
}
```

2. *Fitness function 2:*

```
function Evaluate()
{
  var left = 1*robot.S(0) +1* robot.S(1) + 1*robot.S(2);
  var right = 1*robot.S(5) + 1*robot.S(4) + 1*robot.S(3);
  var result = 0;

  if (right > 1023)
    result += 1000 - right;

  else if (right < 20)
    result += (1000/20) * right;

  else
    result += 1000;

  if (left > 1023)
    result += 1000 - left;

  else if (left < 20)
    result += (1000/20) * left;
```

```
            else
              result += 1000;

            result += 100 * (Math.abs(robot.M1) + Math.abs(robot.M2) -
      (Math.abs(robot.M1 - robot.M2)));

            return result;
}
```

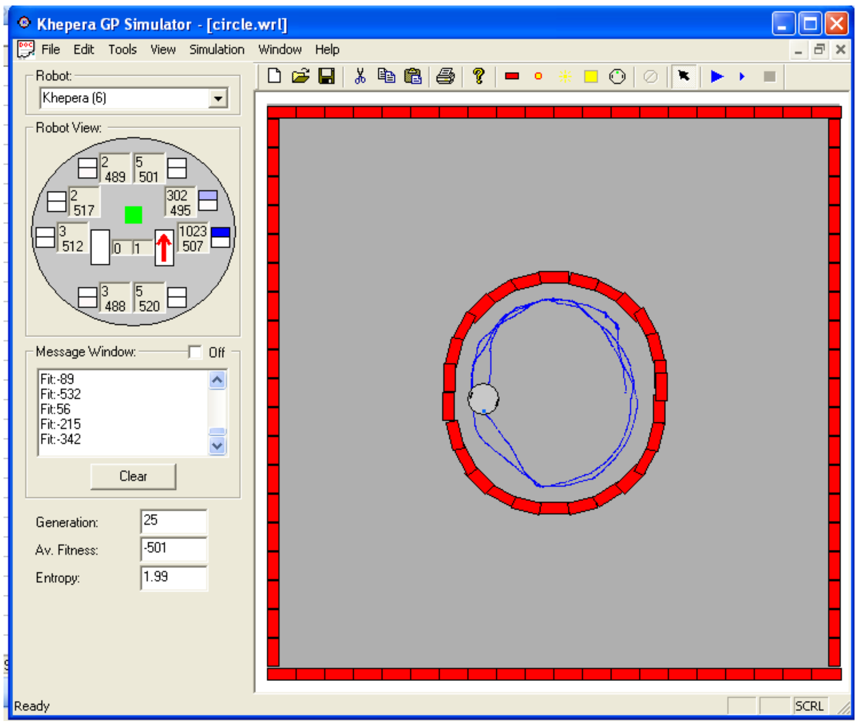
3. *Fitness function 3:*

```
          function Evaluate()
          {
            var sens =  1*Math.abs(robot.S(0) - 1000) +1*Math.abs(robot.S(1) - 500) +
      1*Math.abs(robot.S(2));
            var mots =   10 * (Math.abs(robot.M1) + Math.abs(robot.M2) -
      (Math.abs(robot.M1 - robot.M2)));

          return mots - sens;
}
```

# APPENDIX A2

***Screenshots of learning experiments.***

1. *Training in the circle:*

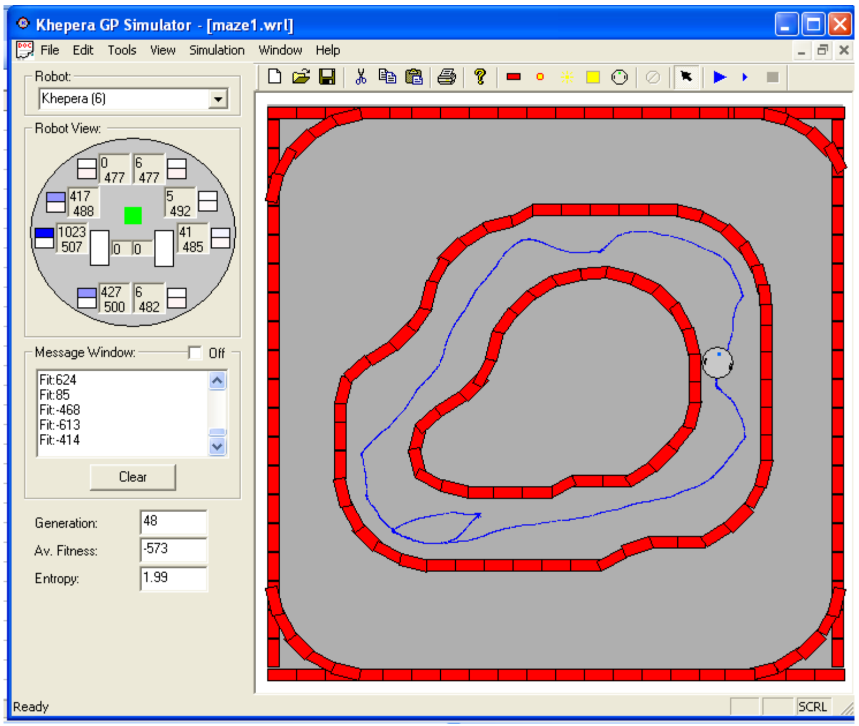


Robot: Khepera (6)

```
Fit:-89
Fit:-532
Fit:56
Fit:-215
Fit:-342
```

Generation: 25
Av. Fitness: -501
Entropy: 1.99

Training phase



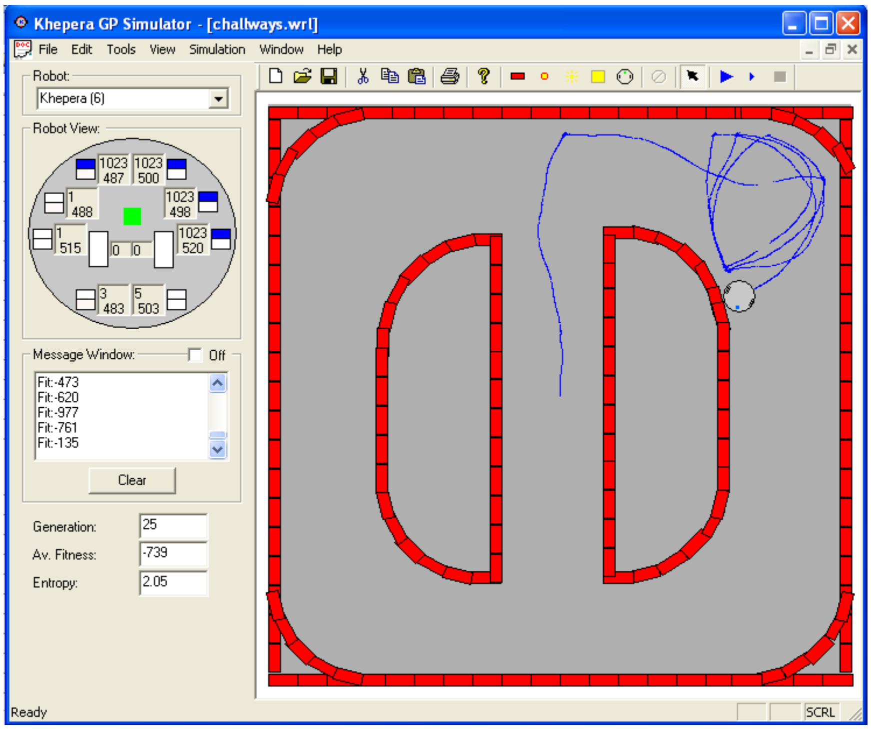
Robot: Khepera (6)

```
Fit:624
Fit:85
Fit:-468
Fit:-613
Fit:-414
```

Generation: 48
Av. Fitness: -573
Entropy: 1.99

Navigating in the maze phase

2. *Training in "chalways" environment:*



Khepera GP Simulator - [challways.wrl]
File Edit Tools View Simulation Window Help
Robot View:
1023 487
1023 500
1 488
1023 498
1 515
0
0
1023 520
3 483
5 503
Off
Clear
SCRL

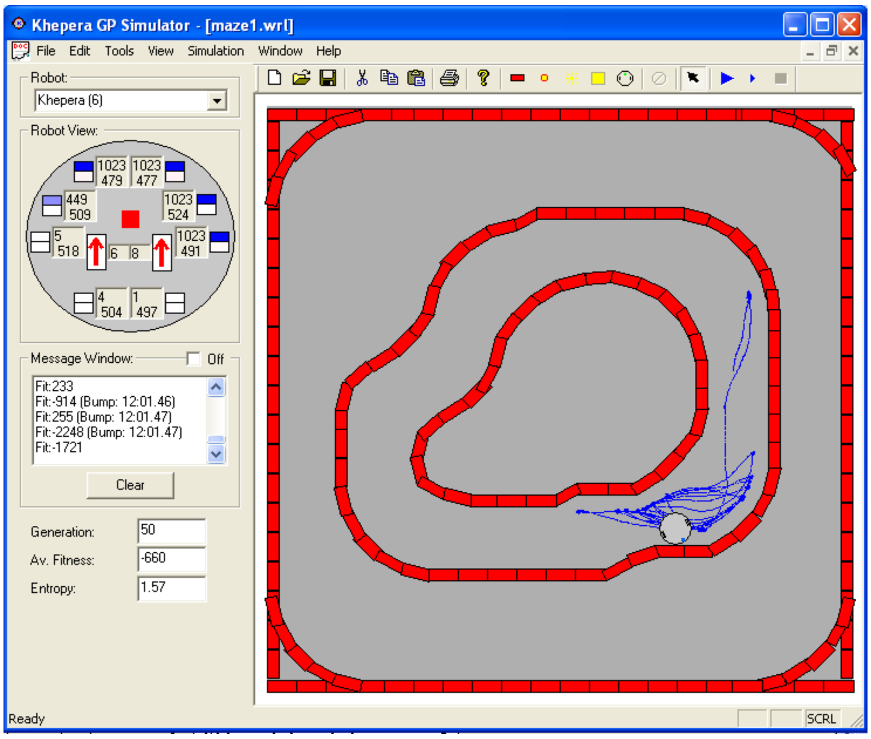
Robot: Khepera (6)

Message Window:
Fit:-473
Fit:-620
Fit:-977
Fit:-761
Fit:-135

Generation: 25
Av. Fitness: -739
Entropy: 2.05

Ready

Training phase



Robot: Khepera (6)

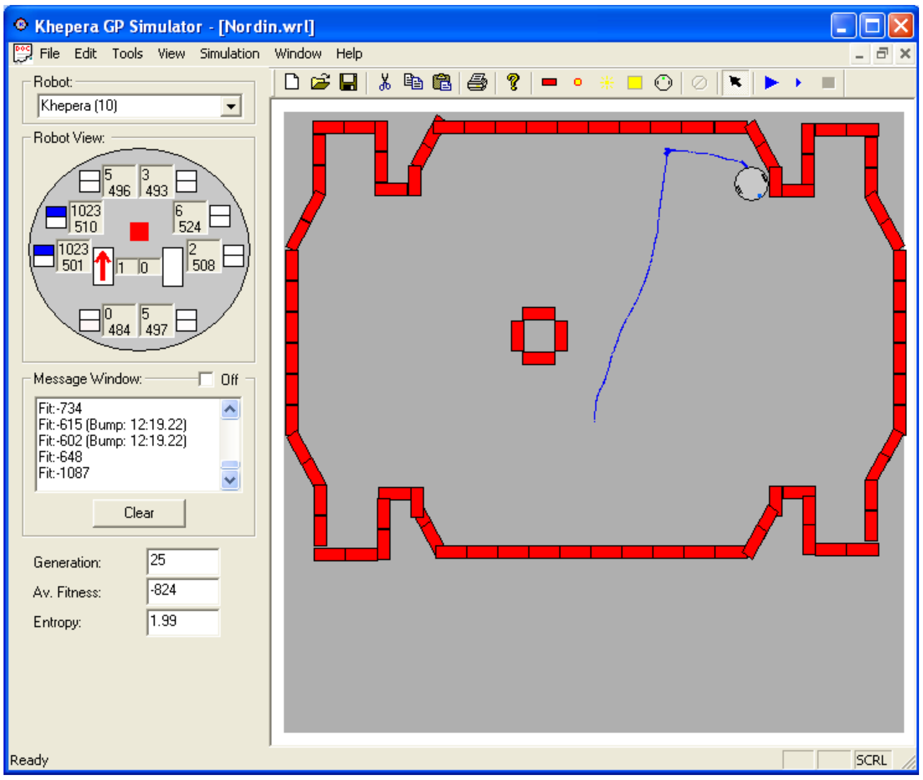
Message Window:
Fit:233
Fit:-914 (Bump: 12:01.46)
Fit:255 (Bump: 12:01.47)
Fit:-2248 (Bump: 12:01.47)
Fit:-1721

Generation: 50
Av. Fitness: -660
Entropy: 1.57

Ready

Navigating in the maze phase

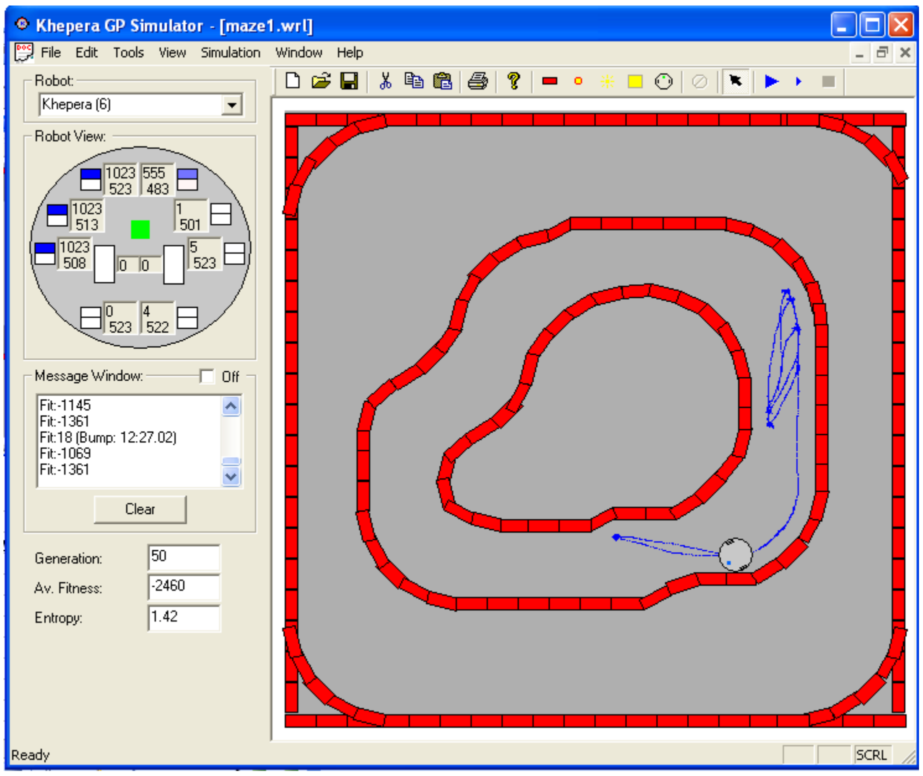
*3. Training in "Nordin" environment:*


Khepera GP Simulator - [Nordin.wrl]
File Edit Tools View Simulation Window Help
Robot:
Khepera (10)
Robot View:
Message Window:
Off
Clear
Ready
SCRL

Fit:-734
Fit:-615 (Bump: 12:19.22)
Fit:-602 (Bump: 12:19.22)
Fit:-648
Fit:-1087

Generation: 25
Av. Fitness: -824
Entropy: 1.99

Training phase

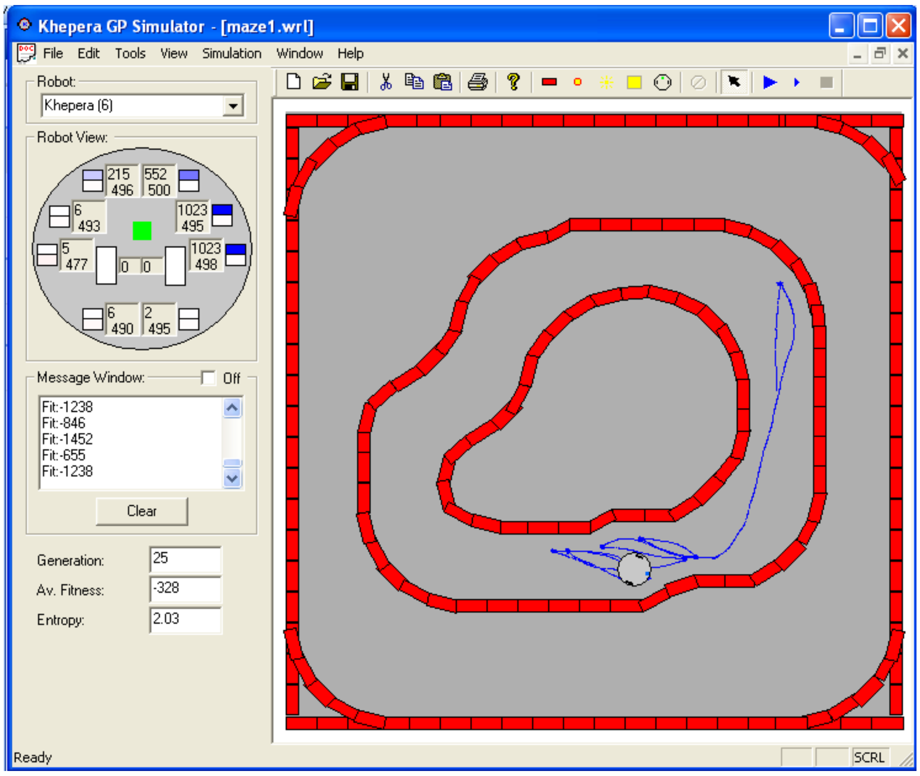

Khepera GP Simulator - [maze1.wrl]
File Edit Tools View Simulation Window Help
Robot:
Khepera (6)
Robot View:
Message Window:
Off
Clear
Ready
SCRL

Fit:-1145
Fit:-1361
Fit:18 (Bump: 12:27.02)
Fit:-1069
Fit:-1361

Generation: 50
Av. Fitness: -2460
Entropy: 1.42

Navigating in the maze phase

4. *Training in maze environment:*


Khepera GP Simulator - [maze1.wrl]
File Edit Tools View Simulation Window Help
Robot View:
215 496
552 500
6 493
1023 495
5 477
0
0
1023 498
6 490
2 495
Clear
Ready
SCRL

Robot: Khepera (6)

Message Window: Off

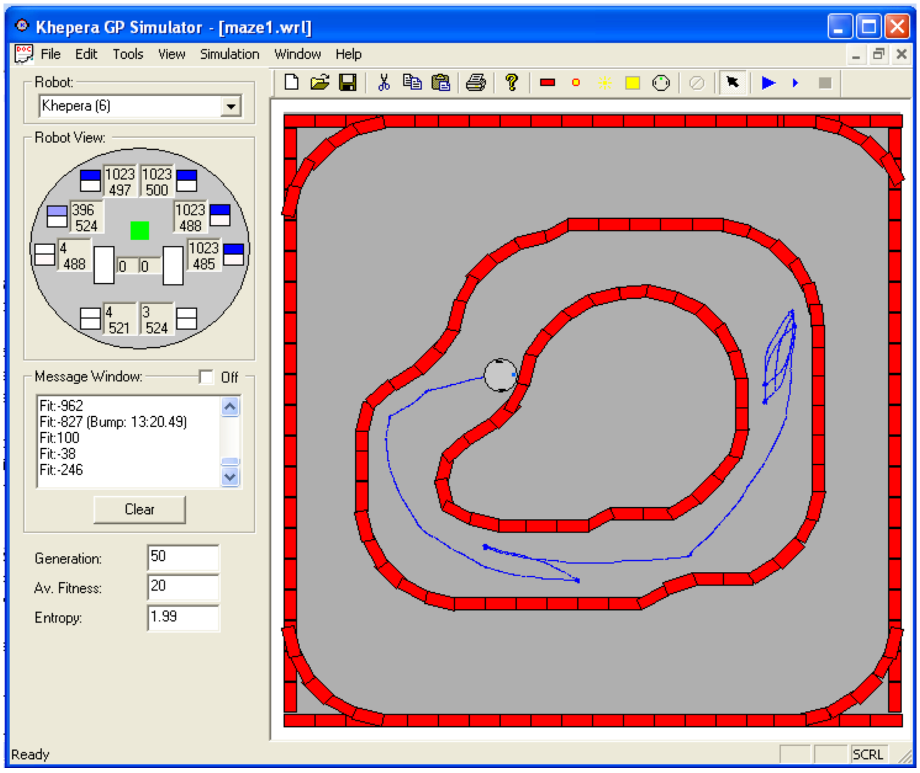
```
Fit:-1238
Fit:-846
Fit:-1452
Fit:-655
Fit:-1238
```

Generation: 25

Av. Fitness: -328

Entropy: 2.03

Training phase



Robot: Khepera (6)

Message Window: Off

```
Fit:-962
Fit:-827 (Bump: 13:20.49)
Fit:100
Fit:-38
Fit:-246
```

Generation: 50

Av. Fitness: 20

Entropy: 1.99

Navigating in the maze phase