TECHNICAL UNIVERSITY OF KOŠICE
FAKULTY OF ELECTRICAL ENGINEERING AND INFORMATICS

FUZZY COGNITIVE MAPS FOR CONTROL TASKS IN INTELLIGENT SPACE

Dissertation

2018 **Ing. Michal Puheim**

TECHNICAL UNIVERSITY OF KOŠICE
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS

FUZZY COGNITIVE MAPS FOR CONTROL TASKS IN INTELLIGENT SPACE

Dissertation

Study program:	Intelligent systems
Field of study:	Cybernetics
Department:	Department of Cybernetics and Artificial Intelligence
Supervisor:	doc. Dr. Ing. Ján Vaščák

2018 Košice **Ing. Michal Puheim**

Abstract

The main subject of the proposed dissertation is an application of fuzzy cognitive maps for control of complex systems, specifically the intelligent space based on a paradigm of the internet of things. The dissertation provides a review of various methods and tools applicable to control complex systems. The first part of the dissertation introduces the methodology of situational control applicable to control complex systems using the means of artificial intelligence, with a focus on fuzzy cognitive maps. In the next part, extensions to basic fuzzy cognitive maps are proposed and implemented using a novel programming library with its appropriate web-based application programming interface. And finally, this interface is used to prototype a specific model of situational control of intelligent space within a scope of a mobile robot navigation.

Keywords

Fuzzy Cognitive Maps, Situational Control, Intelligent Space, Robot Navigation

Abstrakt

Hlavným predmetom predloženej dizertačnej práce je využitie fuzzy kognitívnych máp pri riadení zložitých systémov, konkrétne inteligentného priestoru založeného na paradigme internetu vecí. Dizertačná práca poskytuje prehľad rozličných metód a nástrojov aplikovateľných pri riadení zložitých systémov. Prvá časť dizertačnej práce uvádza metodiku situačného riadenia použiteľnú pri riadení zložitých systémov s využitím prostriedkov umelej inteligencie, so zameraním na fuzzy kognitívne mapy. V ďalšej časti sú predstavené rozšírenia základných fuzzy kognitívnych máp, ktoré sú následne implementované s použitím novo-vytvorenej programovej knižnice pomocou príslušného aplikačného programového rozhrania. Toto rozhranie je nakoniec použité na vytvorenie prototypu konkrétneho modelu situačného riadenia inteligentného priestoru v rámci úlohy navigácie mobilného robota.

Kľúčové slová

Fuzzy kognitívne mapy, situačné riadenie, inteligentný priestor, navigácia robota

Declaration

I hereby declare that this thesis is my own work and effort. Where other sources of information have been used, they have been acknowledged.

Košice, October 29, 2018 .....

signature

Acknowledgement

I would like to express a sincere gratitude to my former supervisor Dr.h.c. mult. prof. Ing. Ladislav Madarász PhD. for his patience, care and diligence during initial years of my PhD study. May he rest in peace. Similarly, many thanks to my colleagues from all collaborating departments at the Technical University in Košice. Last but not least, I would like to thank to my family for their endless support.

Table of Contents

List of Figures 9
List of Tables 12
List of Symbols and Abbreviations 13
Introduction 15
1. Control and Modeling of Complex Systems 17
1.1. Complex System 17
1.2. Fundamentals of Control and Modeling 18
1.3. Control and Modeling of Complex Systems 20
1.4. Situational Control and Modeling of Complex Systems 22
1.4.1. Methodology of Situational Control 22
1.4.2. Situational Modeling 25
2. Computational Intelligence for System Control and Modeling 26
2.1. Overview of Computational Intelligence 27
2.1.1. Artificial Neural Networks 28
2.1.2. Evolutionary and Population Based Optimization 31
2.1.3. Fuzzy Systems 33
2.2. Hybrid Control Methods 37
2.2.1. Hybrid Control of Technical Systems 38
2.2.2. Hybrid Intelligent Systems 39
2.3. Utilization of AI for Situational Control and Modeling 44
3. Fuzzy Cognitive Maps 48
3.1. Cognitive Maps 49
3.2. Fuzzy Cognitive Maps 50
3.3. Formal Definition of FCM 51
3.4. Concept Activation and Inference in FCM 52
4. System Control and Modeling using Fuzzy Cognitive Maps 54
4.1. Basic Problems Solvable with FCM 54
4.2. Modifications of FCM for Control of Complex Systems 55
4.2.1. Disadvantages of Simple FCM 56
4.2.2. Established FCM Extensions 57
4.2.3. Three-Term Relation Neuro-Fuzzy Cognitive Maps 58
4.3. Automated Learning of FCM 65
4.3.1. Interactive Evolution of Fuzzy Cognitive Maps 66

4.3.2. Semi-Interactive Evolution of Fuzzy Cognitive Maps 70
5. Intelligent Space as Complex System 75
5.1. Internet of Things 75
5.1.1. Communication in IoT 76
5.1.2. Information Processing of IoT 81
5.1.3. Applications of IoT 84
5.2. Intelligent Space 86
5.2.1. IoT Devices in Intelligent Space 86
5.2.2. Example Proposal for Intelligent Space 88
5.2.3. Applications in Intelligent Space 91
5.2.4. Ubiquitous Robotics in Intelligent Space 95
5.3. Intelligent Space as Complex System 97
6. Software Tools for Control of Intelligent Space using FCM 99
6.1. Overview of Existing FCM Tools and Frameworks 99
6.1.1. Brief Description of Available FCM Tools and Frameworks 100
6.1.2. Categorization of Available FCM Tools 101
6.1.3. Deficiencies of Available FCM Tools 102
6.2. Open Fuzzy Cognitive Maps Library 102
6.2.1. Library Requirements 102
6.2.2. Preliminary Library Proposal 103
6.2.3. Preliminary Implementation in .NET framework 105
6.2.4. Final Library Proposal 107
6.2.5. Final Implementation in Python 110
6.3. Web Application Programming Interface 114
6.4. User Interfaces 116
7. Fuzzy Cognitive Maps for Control of Intelligent Space 118
7.1. Situational Control of Intelligent Space 118
7.1.1. Description and Global Objectives of Intelligent Space 118
7.1.2. Proposal for Situational Decomposition of Intelligent Space 119
7.1.3. General Strategy for Control of Intelligent Space 121
7.2. Navigation of Mobile Robot in Intelligent Space 124
7.3. Fuzzy Cognitive Maps in Robot Control 125
7.4. Proposal for Navigation System in Intelligent Space 127
7.5. Experiments in Intelligent Space Model 129
Conclusion 134

References 136
Appendices 146
Appendix A – Resumé 147
Appendix B – References to software and documentation 152
Appendix C – Publications of the author 157
Appendix D – Author's CV 161

List of Figures

Fig. 1 Open (non-feedback) control circuit. 20
Fig. 2 Closed (feedback) control circuit. 20
Fig. 3 Control of complex system 24
Fig. 4 Situational model 25
Fig. 5 Structure of artificial neuron. 28
Fig. 6 Multi-layer feed-forward neural network with two hidden layers. 29
Fig. 7 General scheme of evolutionary algorithm. 31
Fig. 8 Exemplary depiction of the fitness landscape 32
Fig. 9 Fuzzy set. 33
Fig. 10 Fuzzy linguistic variable. 34
Fig. 11 Visualization of the inference of fuzzy rules in Mamdani FIS. 35
Fig. 12 Structure of fuzzy controller. 37
Fig. 13 Architecture of a hybrid control system 39
Fig. 14 Hybridization of computational intelligence 40
Fig. 15 NARA – Neural-network designed on Approximate Reasoning Architecture. 42
Fig. 16 Iterative rule generation using AdaBoost algorithm. 44
Fig. 17 Situational control using means of computational intelligence 45
Fig. 18 Formatter control 46
Fig. 19 Simple cognitive map. 49
Fig. 20 Cognitive map of political situation in Lebanon 50
Fig. 21 Fuzzy cognitive map. 51
Fig. 22 Calculation of the activation of concept C_j 52
Fig. 23 Model of heat exchanger realized using FCM 54
Fig. 24 Visualization of three various attractors of the same FCM with two concepts selected. 55
Fig. 25 A visualization of the concept relations within the FCM model of a city. 57
Fig. 26 FCM extended by nonlinear (α), conditional (β) and time-delayed (γ) weights 58
Fig. 27 Example of unwrapping of a simple FCM 60
Fig. 28 A proposal for nonlinear relations between concepts using the FF-ANN 60
Fig. 29 Visualization of an expansion of the concept update formula 62
Fig. 30 A block diagram of a PID controller and a controlled system 63
Fig. 31 An example of the three-term relation between concepts C_i and C_j 64
Fig. 32 Interactive evolutionary optimization. 67

Fig. 33 Example of graphical user interface used for interactive evolution of FCM controller. 68
Fig. 34 Final movement trajectories for the best controller designs: 70
Fig. 35 Evolution of the Internet of Things 76
Fig. 36 Comparison of OSI and TCP/IP models including applicable protocols 79
Fig. 37 Cloud computing services compared to the traditional IT 81
Fig. 38 Typical applications supported by cloud computing model 82
Fig. 39 IoT data processing layers (cloud, fog and edge) 83
Fig. 40 Traffic congestion overlay in Google Maps. 85
Fig. 41 Sensor and actuator cooperation. 87
Fig. 42 Floor plan of the Intelligent Space at CIT 88
Fig. 43 Monitoring system of the Intelligent Space at CIT 89
Fig. 44 Robotic hardware utilized in IS at CIT 89
Fig. 45 Networking Proposal of IS at CIT. 91
Fig. 46 Virtual model of an intelligent household 92
Fig. 47 Proposal of speech emotion recognition & text transcription system 93
Fig. 48 Description of intelligent spaces 94
Fig. 49 Diagram of the home automation system Jarvis 95
Fig. 50 Visit notification using facial recognition via Pepper robot 96
Fig. 51 MATLAB/Simulink model of a simple FCM with three concepts 101
Fig. 52 System diagram of the preliminary proposal for FCM library 104
Fig. 53 Graphical user interface for OpenFCM Library 106
Fig. 54 System diagram of the final proposal for FCM library. 108
Fig. 55 Class diagram of the final implementation of PyOpenFCM library 111
Fig. 56 Efficient representation of a graph in form of an adjacency list. 112
Fig. 57 Online FCM service provided by the library. 114
Fig. 58 Examples of web API usage in a web browser. 115
Fig. 59 Command line interface (CLI) of the online FCM service. 116
Fig. 60 Graphical user interface (GUI) of the online FCM service. 117
Fig. 61 Proposal for situational decomposition of IS. 120
Fig. 62 Proposed model for control of IS. 122
Fig. 63 Sampling of a path into waypoints 125
Fig. 64 Three-term FCM controller of a wheeled mobile robot. 126
Fig. 65 Location analyzer based on FCM with neural relations 127
Fig. 66 Components of a navigation system within IS. 128

Fig. 67 A simulated model of IS 130
Fig. 68 An evaluation GUI for semi-interactive evolutionary optimization 132
Fig. 69 An overview of simulation of a situational control strategy for navigation 133

List of Tables

Tab. 1 Qualitative comparison of AI methods 40
Tab. 2 Relative comparison of experimental results of FCM and AN-FIS controllers 69
Tab. 3 Results of the non-interactive evolution of FCM controller 73
Tab. 4 Results of the fully-interactive evolution of FCM controller 73
Tab. 5 Results of the semi-interactive evolution of FCM controller 73
Tab. 6 Relative comparison of results of all IEO methods 73
Tab. 7 Open Systems Interconnection (ISO/OSI) model 78
Tab. 8 Estimated dataflow from various sensors 90
Tab. 9 Intelligent Space compared to properties of Complex System 98

List of Symbols and Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
BN	Bayesian Network
BP	Backpropagation of Error
CM	Cognitive Map
CS	Complex System
DDHL	Data Driven Hebbian Learning
DLL	Shared Dynamic Library
DS	Dynamic System
EA	Evolutionary Algorithms
FCM	Fuzzy Cognitive Map
FIS	Fuzzy Inference System
FL	Fuzzy Logic
FS	Fuzzy Systems
GB	Giga Binary Byte (230 Bytes)
Gb	Giga Binary Bit (230 Bits)
HL	Hebbian Learning
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IRL	Iterative Rule Learning
IS	Intelligent Space
JRE	Java Runtime Environment
JSON	JavaScript Object Notation (text data interchange format)
KB	Kilo Binary Byte (210 Bytes)
Kb	Kilo Binary Bit (210 Bits)
LMS	Least Mean Square
LV	Linguistic Variable
MB	Mega Binary Byte (220 Bytes)
MF	Membership Function (in Fuzzy Sets)
MISO	Multiple-Input Single-Output function
MRC	MATLAB Runtime Components
MSAGL	Microsoft Automatic Graph Layout

Mb	Mega Binary Bit (220 Bits)
NAT	Network Address Translation
NFCM	Neuro-Fuzzy Cognitive Map
NIST	National Institute of Standards and Technology
PN	Petri Net
SC	Situational Control
SISO	Single-Input Single-Output function
SpMxV	Sparse Matrix-Vector Multiplication
TB	Tera Binary Byte (240 Bytes)
TSK	Takagi-Sugeno-Kang Fuzzy Inference System
TTR	Three Term Relations
Tb	Tera Binary Bit (240 Bits)
UML	Unified Modeling Language
URL	Uniform Resource Locator
WPF	Windows Presentation Foundation
WSGI	Web Server Gateway Interfaces (for Python programming language)
www	World Wide Web

Introduction

"Making a decision was only the beginning of things. When someone makes a decision, he is really diving into a strong current that will carry him to places he had never dreamed of when he first made the decision."

Paulo Coelho

Complex systems are part of our everyday lives. These systems include the majority of social, psychological, biological, technical, physical and other systems that are based on the principle of self-regulation or are controlled by external interventions. An example of such system is an *intelligent space* as a very current trend in proliferation of the *internet of things*. Constant growth and difficulty of addressed problems, including the need to manage increasingly complex systems, inevitably requires the implementation of *new methods* for design and control of these systems. The aim is to bring higher safety, quality and robustness to the control.

This dissertation provides an overview of the methods, procedures and tools that are useful for solving these problems. These include the methodology of *situational control* (referred to in chapter 1), which is applicable to control an extensive range of various complex systems. *Computational intelligence*, currently one of the fastest evolving areas of informatics and computing, also offers the means to achieve these goals (the outline of its methods is shown in chapter 2). One of its methods are *fuzzy cognitive maps* (presented in chapter 3), which are very suitable for description, modeling and control of complex systems. The design of advanced *methods for modeling and adaptation* of fuzzy cognitive maps is also one of the objectives of this work (in chapter 4). In order to test all of the proposed methods, we have proposed the intelligent *control of an intelligent space*, a specific manifestation of the internet of things (presented in chapter 5). According to its attributes it is very suitable for testing and verification of various models and control algorithms, which can be later generalized and applicable in other areas. Since individual models need to be created using *specific programming tools*, we have proposed and implemented a *novel programming library* and a web-based *application programming interface* (described in chapter 6) specifically oriented towards building models with fuzzy cognitive maps. And finally, we have used this library in order to implement a *specific control strategy* (introduced in chapter 7) applicable to *situational frame* of *robot navigation* within the intelligent space.

Goals and contributions of the dissertation were formalized accordingly, and defined as these tasks:

1) *To modify the basic concept of fuzzy cognitive maps (FCM) in order to allow modeling and control of complex systems (scientific goal).*
2) *To create a programming library for FCM computations applicable to modeling and control of complex systems, including auxiliary graphical user interface (GUI) and application programming interface (API) (technological goal).*
3) *To propose a situational control model of an intelligent space utilizing FCM (scientific goal).*
4) *To propose a control structure of an intelligent space for a situational class of robot navigation (scientific goal).*
5) *To create a simulation of a situational control of elements of an intelligent space (technological goal).*

The *first goal* is accomplished in chapter 4 where we introduce several *modifications of FCMs* which simplify modeling of complex systems (ch. 4.2) along with *learning algorithms* that enable automated adaptation of FCM models (ch. 4.3).

The *second goal* is fulfilled in chapter 6 in which we present a proposal and implementation of a novel *FCM library* (ch. 6.2) along with web-based *application programming interface* (ch. 6.3) and corresponding *graphical user interface* (ch. 6.4).

The *third goal* is achieved in chapter 7.1 where we propose a *general description and global objectives* of the intelligent space (ch. 7.1.1) along with a proposal for *situational decomposition* (ch. 7.1.2) and a *general control structure* (ch. 7.1.3) based on a *formatter* consisting of several network controllers and analyzers which can be implemented as FCMs.

The *fourth goal* is accomplished mostly in chapter 7.4. However, partial problems are discussed in previous chapters, including general *description of navigation* of a mobile robot (ch. 7.2) and a particular implementation of *FCM-based controllers* and *analyzers* (ch. 7.3).

The solution to the *last goal*, which is the *simulation* of the selected elements of intelligent space required to perform *robot navigation*, is presented in chapter 7.5.

1. Control and Modeling of Complex Systems

"The knowledge of the immense complexity of reality, is a matter of respect for all life, inspiring us into awe."

Karel Čapek

In control of complex systems that contain a large number of elements (technical, non-technical means and people), we are often unable to create a precise mathematical model of the controlled system. Even if we were able to construct it, it would probably be very complex and difficult to manage practically. However, in order to design and control such a system, we may use the methodology of s*ituational control* that will decompose the control of the system into minor problems, depending on the current state of the system. These minor problems can be solved using available algorithms and technology [1].

1.1. Complex System

An arbitrary system may be considered as "purposely defined set of elements and links between them, which is separated from other objects in space and time by a real or virtual boundary" [2]. It is already clear from this definition that a particular system and its characteristics may vary considerably depending on the different baseline requirements. For this reason, the concept of a *complex system* (CS) can be attributed to a wide range of controlled objects and relevant control subsystems. The mere term "complex system" is not unique. In very close and often confused with terms such as "large system", "extensive system", "complex" and other. The exact definition of the concept is challenging and different approaches are used by various authors [3].

For our purposes we will use the definition [4], according to which the CS consists of the final number of elements (systems, subsystems, elements) and their interconnections, which are subject to the following characteristics [1][3][4][5][6][7]:

- There is a *large number of elements* that can be readjusted (the number of elements in the CS may increase or decrease).
- *Relationships between the elements* defining the structure of the CS may change as well (including the elements themselves).
- *Hierarchy of the organizational structure* of the components of the CS (regardless of whether it is a management or controlled system).
- The right of *autonomous decision-making of elements* (the organization of the CS is often decentralized, so that individual elements have large degree of autonomy).
- *Change of space* (dimension of space) in which the CS is located and in which it operates.

- *Change in the organizational structure* of the CS (i.e. changing links or relationships between elements; links and relationship of elements of the CS may be changed through organizational interventions in the course of its management in order to fulfill the predetermined global management objectives).

At the same time, a number of different side effects can influence the behavior of the CS, e.g. the disruptive effects of the surrounding area (the faults), the effects of disruptive communication between the elements (the noise), the chaotic behavior of the environment and the system itself, the human factor (in ergatic systems) and other accidental influences. The unpredictability of system behavior therefore requires the ability to influence its structure depending on the current situation, i.e. the ability to reorganize, restructure and adapt both elements of the CS and the corresponding management mechanisms [5]. From the characteristics of the CS mentioned in the previous paragraphs, it is clear that the structure of the CS can be managed by [1]:

- influencing the individual elements of CS,
- influencing the relations between the elements of CS.

In the case of an effort to change the structure of CS, it is an important prerequisite to be able to describe and display it. To this end, two basic approaches [1] can be used in principle:

- graphs and graph theory,
- structural, interaction (binding) matrices.

The description through graphs can be particularly advantageous in terms of visualization, although in a large number of related elements, clarity is lost to some extent. On the other hand, the registration using the interaction matrices is useful for machine processing.

1.2. Fundamentals of Control and Modeling

Modelling of dynamic system can be defined as the process of making a system description or the process of determining its qualitative and quantitative characteristics [8]. For this purpose, we use mathematical models that involve the interaction of individual system agents, elements or subsystems, as well as their interaction with the environment. The interaction between the elements can be expressed by direct interconnection of the elements or indirectly as the joint action towards another subsystem (in case of co-operation of the elements) or in both ways [8]. It is important to note that the mathematical model describing the system and the relationship between its elements is always only a purposeful abstraction of the real system and its accuracy is limited, because in order to cope with the actual complexity of the system it is often necessary to

carry out various simplification [3]. Common forms of the description of both linear and nonlinear systems are mainly [8][9]:

- *Linear differential equations*,
- *Laplace transform*,
- *Fourier transform*,
- *Transient, impulse and frequency characteristics*,
- *State Space methods*,
- *Lagrange equations*,
- *Neural Networks*,
- *Fuzzy-Neural models*.

For the control of the dynamic systems, we consider a purposeful action designed to influence the dynamic processes between the elements of the system so that specific pre-defined control objectives are achieved [8]. The basic management objectives include [8]:

- *Stabilization* – where the objective is to achieve the required stable value of the regulated variable (e.g. regulation of the water level in the tank),
- *Dependent control* (towed regulation) – where the control target is derived from the sensor-obtained parameter of another independent system (e.g. an electrically adjustable central heating control),
- *Extremal regulation* (optimization) – aimed at maximizing or minimizing the criterial function, whose parameters can be the input-output variables of the system and its internal states.

Depending on how the system model information is used or depending on the output regulated variables, we divide the basic types of control into [8]:

- *Non-feedback control* (direct or open control) – where we do not use the response of the controlled system to the control actions; A prerequisite is the knowledge of a sufficiently accurate (inverse) model of the controlled system, which allows pre-assessment of the impact of the chosen control actions; In particular, open (or compensatory) control circuits are used in this type of control (see Fig. 1);

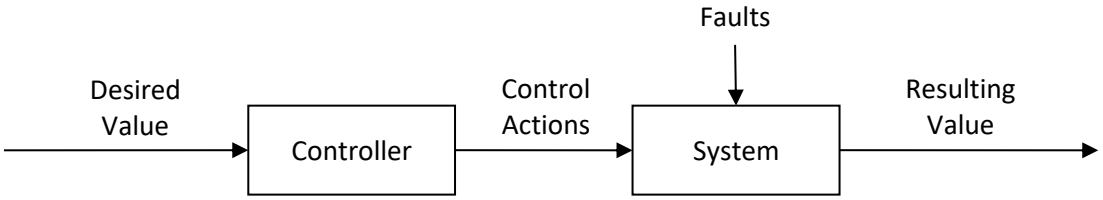


Fig. 1 Open (non-feedback) control circuit.

- *Feedback control* – where the exact model of the system is not known, but by monitoring the deviation of the current value of the selected system parameter (or parameters – current inputs, outputs and/or states) from the target value, we can compute the required action interventions with sufficient precision; In determining the deviation from the required value, closed (feedback or regulatory) control circuits are used (see Fig. 2);

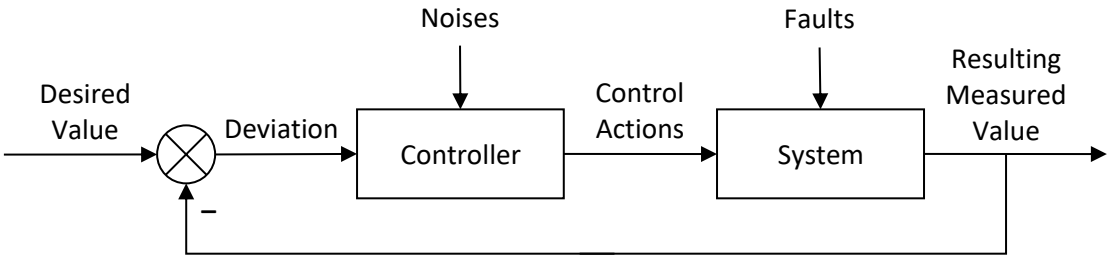


Fig. 2 Closed (feedback) control circuit.

- *General control* – which is a combination of the previous two types; In this type of control, combined control structures are used, drawn from both open and closed control circuits.

A taxonomic classification with regard to the participation of the human agent in the process of managing the system is very common as well. Accordingly, we distinguish [8]:

- *Automatic control* with absence of human intervention,
- *Automated control* with partial human intervention.

1.3. Control and Modeling of Complex Systems

Complex systems are described and we determine their properties (as with other dynamic systems) using mathematical models. Complex multi-parametric systems can be created by combining several sub-systems. They often have a hierarchical structure, the main causes of which are [7]:

- Large dimension of controlled CS;
- Varying awareness of the various elements of the CS about the impact of the surrounding environment on the CS and about the parameters of the controlled process.

Traditional methods and procedures designed to analyze systems and design control strategies are based on the general premise of the centrality where system information and calculations based on it are centralized. This assumption usually fails in control of CS due to a lack of centralized information or for insufficient computing power of the centralized system [3]. A common feature of CS is the transfer of part of the decision-making process from the center to individual subsystems arranged in a decentralized, distributed or hierarchical structure. Subsequently, the properties of the CS can be approximated by tracking the properties of these subsystems. The CS with hierarchical structures can be controlled using various approaches [1][3][6][10]:

- ***Decomposition methods of CS control*** – for high-dimensional CS, the decomposition methods can be used, the essence of which is the targeted reduction of the global system to simpler tasks easily solvable by subsystems.
- ***Decentralized control of the CS*** – a fundamental characteristic is the existence of a boundary in the transmission of information between the various subsystems of the CS. For this reason, the center inevitably leaves part of its decision-making to the subsystem. Decentralization, however, raises the need to coordinate individual subsystems in order to achieve a global objective.
- ***Information hierarchical-organization management systems*** – emphasize, in particular, the achievement of a-priori evaluation of the different variants of the hierarchical organizational structure selected for the chosen complex system.
- ***Situational control of CS*** – which, based on the determination of the global state of the controlled CS, selects an appropriate control strategy. The selected control strategy can be used for several system states, as the number of possible states (situations) is usually significantly higher than the number of available control strategies. For a more detailed description of the *Situational control* methodology, see Chapter 1.4.1.
- ***Anytime control of CS*** – these algorithms focus on a specific subset of situational control problems, e.g. control situations in which sampling errors occur, for example due to overload of the sensory system or computing system, or because of the malfunctioning state of the system itself.
- ***Cooperative control of the CS*** – the goal-setting subsystem for the sets out the objectives of the function of CS and lays down the rules for other subsystems and elements. These subsystems have considerable autonomy in their behavior, but they also use mutual cooperation to achieve the common objective.

- ***Chaos theory in control of CS*** – the implications of chaos theory are mainly used to control the chaos contained in the controlled CS.
- ***Means of artificial Intelligence*** – recently used in many areas of control and modeling of (not only) CS. These may include, for example, *expert systems*, *agent* and *multi-agent systems*, *evolutionary* approaches, *artificial neural networks* and *fuzzy systems*, which are further described in Chapter 2.

With regard to the need for a thorough analysis of the structure of the controlled CS and the respective control systems, the process of control the CS is usually divided into two stages [1]:

1) *Planning the control process*,
2) *Operational control*.

Planning of the control process is separate from the operational control (both in time and space). It is based on a set of hypotheses about the dynamics of the CS and the environment within its entire scope of activity. As a result of the planning phase, the methodology and structure of the control system as well as the relevant control algorithms [1] are proposed. The operational control then continuously removes deviations from desired state using the proposed control algorithms, which are suitably parametrized during run-time depending on the current objectives. The optimization of parameters may relate to the quality of decision-making and control processes, or to minimize variances against the plan, but it is almost impossible to optimize the global entire control process [1].

1.4. Situational Control and Modeling of Complex Systems

Situational control (SC) is an important methodological practice applicable to the control of CS [2]. The starting point for using the SC methodology is the assumption that the controlled CS can be found in infinitely large number of states, while the amount of resources enabling its control is usually limited [5]. When using the principle of SC, a formal model is integrated into the control system, which allows to anticipate the consequences of individual decisions and to choose the most appropriate solution with regard to the chosen control criteria [1]. In other words, for the different situations in which the CS may be located, we assign pre-prepared decisions (control strategies), which count is usually much smaller than the number of possible situations [6].

1.4.1. Methodology of Situational Control

The proposal for *situational control of complex system* is a challenging process consisting of several steps. In principle, these steps can be divided (as described in Chapter 1.3) to the *planning and design stage* and subsequently to the *operational control stage* of the CS.

The *planning stage* of the control of CS begins with:

1) a *description of the controlled system*,
2) the *setting of global control objectives*,
3) and a *proposal of situational classes* (frames), which can group several similar situations or states of the CS.

The relevant control strategies are then assigned to the individual situational classes. Each situational class covers at least one control strategy, but there may also be cases where one class covers several strategies. In the event that one specific strategy covers several different situational classes, it is appropriate to consolidate these classes and, where appropriate, to revise the proposal in general. To ensure effective operation of the control system, the number of classes and strategies should be minimized [5].

The role of *operational control* of the CS is the parameterization of pre-prepared control strategies according to the assessment of the state of the CS and their use in system control. The goal is to offset the deviations from the required system state in real time [1]. The process of control of the CS contains three fundamental decision-making nodes (see Fig 2), which, according to the chosen decision-making criteria and the current situation of the CS, make decisions about [1]:

- The classification to the appropriate class of situations (CS state estimation),
- The choice of a corresponding control strategy,
- The realization of particular control functions.

When making decisions, it is possible to make use of the classic principle of decision-making tables, where the rows represent possible situations and the columns show the appropriate decisions [1]. Since the SC is a framework methodology, it can also use other applicable methods from the various areas (listed in a chapter 1.3) to address the individual tasks related to decision-making (e.g. analysis, classification, modeling, etc.).

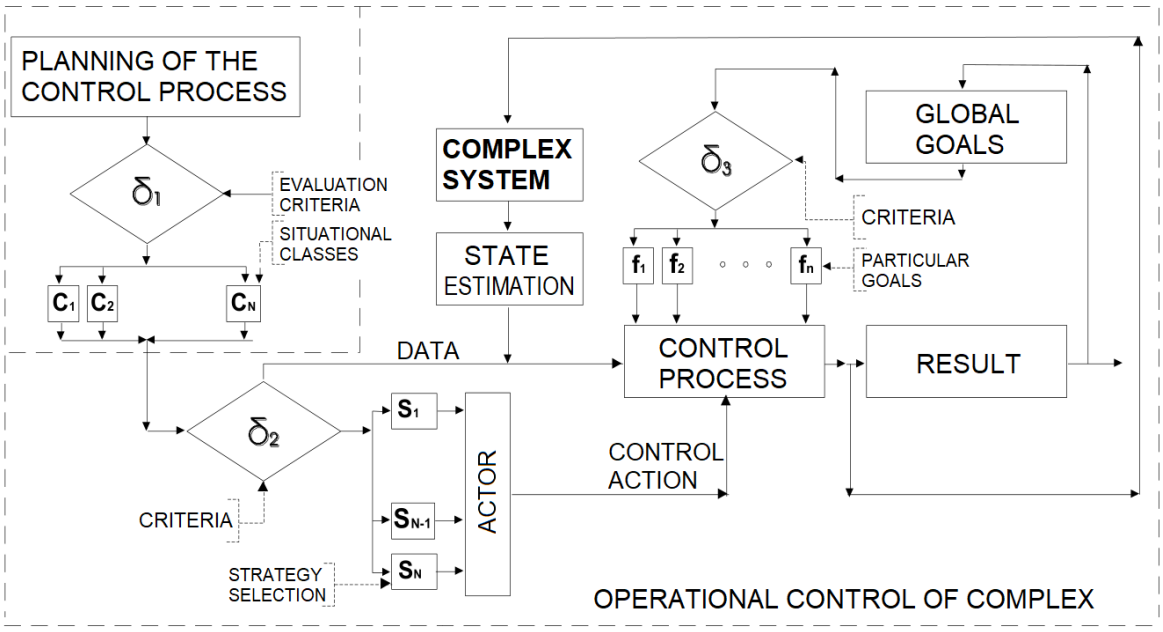


Fig. 3 Control of complex system [11].

It is clear from the above, that in order to use the *situational control* methodology, the following problems need to be solved beforehand [6]:

- How to prepare a set of appropriate and satisfactory decisions in advance,
- How to categorize individual situations into a class of situations so that each class of situations can be assigned to an appropriate decision,
- How to perform the classification of current situations into the classes of situations during the actual operation of the system.

The way to solve these problems is related to the nature of a specific controlled CS. It also depends on the chosen decision-making methods and the CS control proposal. It is possible to use conventional methods (e.g. mathematical-statistical methods, multi-criterial decision-making, group decision-making, heuristic procedures, etc.) or smarter approaches integrating elements of artificial intelligence [1][5]. The algorithm for the creation of a *situational control of CS* is usually made up of the following steps [1]:

- Description of model structure and function of the controlled CS.
- Identifying the global objective.
- Classification of operational states and their causes.
- Classification and description of control strategies assigned to each state,
- Algorithmization of individual control strategies;
- Implementation.

The specific implementation of *situational control*, in our case the *formator control* using the means of *computational intelligence*, is given in chapter 2.3.

1.4.2. Situational Modeling

Although the methods of situational classification were originally intended mainly for use in the control of CS, they can be also applied to the creation of models of dynamic systems [5]. An example is a set of linear models, where each model corresponds to the system behavior at specific work point (state or situation), and the corresponding classifier that determines which of the models is suitable for the current situation. The advantage of this approach is the overall simplification of the system model, which can be modeled with varying accuracy depending on requirements of the situation. If necessary, the model in a particular situation may be further decomposable to more precise local models corresponding to micro-situations at a lower level (see Fig 3).

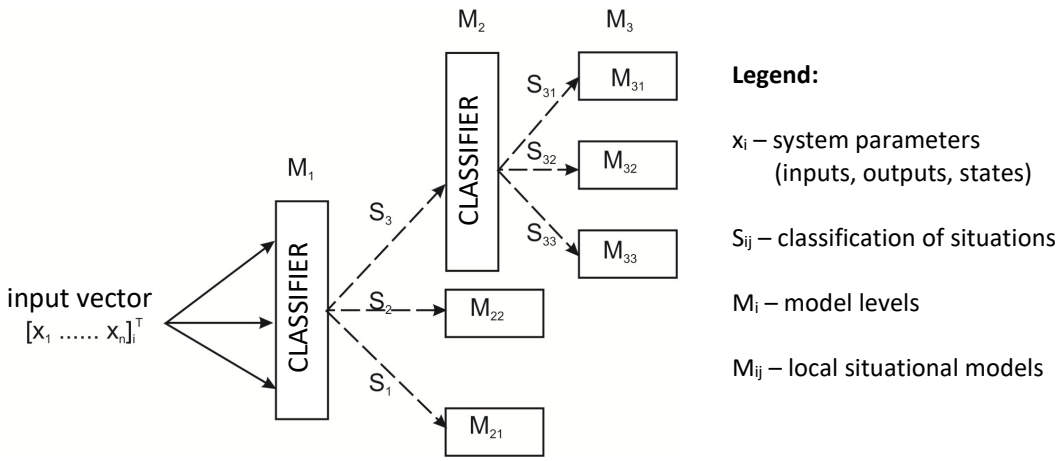


Fig. 4 Situational model [5].

Situational models can be used to predict system behavior, diagnose or test appropriate control strategies. Their great advantage is the ability to model even nonlinear CS in its entirety, which, in the case of traditional models, is very difficult or even impossible.

2. Computational Intelligence for System Control and Modeling

"Every sufficiently advanced technology is indistinguishable from magic."

Arthur C. Clarke

Artificial intelligence (AI) together with informatics, microelectronics and telecommunication technology leads to deep structural changes in all areas of our economic and social lives. The penetration of AI extends from elementary sciences, through operational planning and design of robots to popular culture. AI is not only a set of computational methods that supports informatics and computer science. From its inception, the main goal and inspiration is to study the intelligence as a natural phenomenon [12].

Intelligent technologies are systems that have the ability to deal with tasks that are expected to be solved only by a person with knowledge of the problem domain [13]. The basic features of intelligent technologies include the following capabilities [13]:

- *Learn* from data and gain knowledge.
- *Store knowledge* acquired in the past.
- *Use obtained knowledge* in future problem-solving.

The need for systems capable of learning leads to a proposal of *learning systems* or *intelligent systems*. These systems have the ability to obtain information and knowledge about the individual states of the system and, if necessary, replace the human operator or other control system. The Intelligent System (IS) can obtain knowledge in various ways [13]:

- It can create an arranged *pair of system inputs and outputs*.
- It is able to *classify* system states into *model classes* (even if the system is described by a multidimensional state vector).

In order to ensure the ability to store knowledge, it is essential to build a *knowledge base* of some sort. IS can use the existing human expertise to create the knowledge base and then use it in a similar way as a person would. The quality of the decision-making is determined by the quality of knowledge available to the system. According to its representation, the knowledge can be divided to [1]:

- *Declarative knowledge* that expresses *what* needs to be known or done.
- *Procedural knowledge* that defines *how* to take some action.

In the core of IS there is a control (interpretation, inference) mechanism that uses the knowledge stored within the knowledge base to address a particular problem. The purpose of storing knowledge (acquired on the basis of past experience) is therefore its use in the future. In practice, this means that if an operational situation has been established in the past and the human operator has successfully resolved it with his intervention, the IS with this knowledge should be able to respond to the similar situation in the future automatically and independently [1].

2.1. Overview of Computational Intelligence

The methods of *artificial intelligence* can be divided into two main branches [12]:

- Traditional approaches based on the methods of *symbolic artificial intelligence*, which are characterized by the use of mathematical logic, symbolic representation of knowledge and central sequential processing. They represent a “top-down” approach, which assumes that intelligence is rational and can be represented by logical systems involving evidence of the truthfulness of facts from basic axioms to final consequences.
- Modern approaches based on methods of *computational intelligence* (also known as *subsymbolic artificial intelligence* or *soft computing*), which are characterized by a non-symbolic representation of knowledge. They represent the “bottom-up” approach, which is inspired mainly by low-level perception processes, biological processes, evolutionary principles and natural phenomena. The prevailing philosophy is *connectionism*, which assumes that simple parts can be combined to assemble sophisticated systems with high abstract intelligence.

The main distinction between these methods is obviously in different forms of knowledge representation. The symbolic AI uses simple and compact expressions of speech, symbols and mathematical laws. The subsymbolic AI uses numerical representation, works with real numbers, pictures and acoustic information [12].

In the next part of the work we will pay particular attention to the means of the *computational intelligence*, the main representatives of which are *artificial neural networks*, *fuzzy systems* and *evolutionary algorithms*.

2.1.1. Artificial Neural Networks

An artificial neural network (ANN) is a massive parallel processor that can store knowledge of the solution to the specific problem and use it to solve similar problems in the future [14]. ANN is a recognized way to address a wide range of problems, such as problems of approximation of functions, classification, association problems, memory simulation, signal transition, etc. The wide applicability of the ANN has roots in their ability to approximate any continuous function. This ability is demonstrated by the *universal approximation theorem*, which has been proved independently in [16][17] and [18].

The basic structural and processing unit of the ANN is an artificial neuron (see Fig. 5).

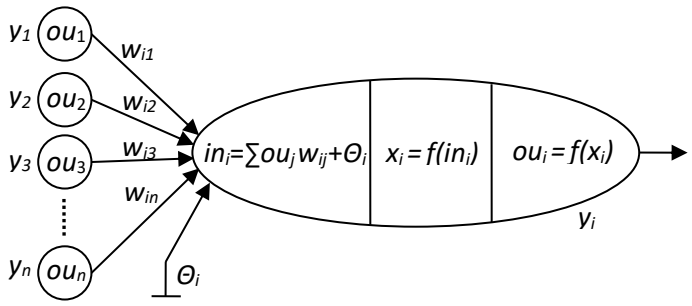


Fig. 5 Structure of artificial neuron.

The neuron influences other neurons through one-way links, *synapses*. The strength of influence between two neurons is directly proportional to the value of the *synaptic weight* w_{ij} corresponding to the link. In addition to the influence of neurons, there is also an influence of the *threshold* (*bias*), which is actually a permanent constant effect independent of the influence of other neurons. Its strength depends on the threshold weight Θ_i. The values of synaptic weights are the main parameters determining the function of the ANN and therefore represent knowledge stored in the ANN [14].

The neuron y_i is activated during its operation in ANN, depending on the influence of the other neurons y_j connected to it. Its activation ou_i usually takes values in range from 0 to1. If we define neurons y_j by using their output values ou_j as $y_j = ou_j$ and the activation function as φ, then the output of the neuron y_i can be defined simply as:

$$y_i = \varphi\left(\sum_{j=1}^{N} w_{ij} y_j + \Theta_i\right). \tag{1}$$

Artificial neurons in the ANN are organized into more complex structures with various arrangement. In general, the ANN can have arbitrary structure describable by an oriented graph, but the

characteristics of such networks are difficult to analyze. It is more common to use the ANN with regular structures, e.g. multi-layer structure with one input layer, one output layer and one or more hidden layers. In general, we may divide the topology of the ANN into two basic groups [14][15]:

- Feed-forward (FF ANN) – where neurons are connected in one direction only.
- Recurrent (RC ANN) – which allow the spread of the signal in both directions and the distribution of neurons to the layers is often ambiguous.

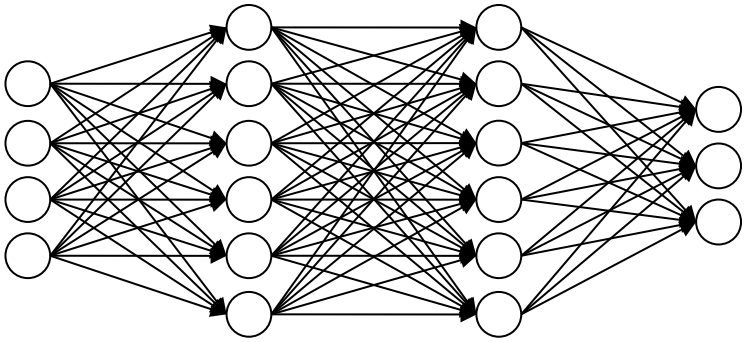


Fig. 6 Multi-layer feed-forward neural network with two hidden layers.

The operation of the ANN can be divided into two phases [14]:

- *Learning phase* (adaptation) – during which the knowledge is stored in the ANN, through a goal-oriented process of changing the synaptic weights.
- *Life phase* – during which the knowledge obtained in the learning process will be used to address a problem (e.g. classification, optimization, aggregation, etc.). In this phase, the synaptic weights are no longer changing.

The learning of the ANN can be done by two fundamental approaches. The first one is *supervised learning* (sometimes referred to as learning with a teacher), where the corresponding output data is available for the input data. The results of the ANN propagation during the learning process can therefore be compared with the reference data. The second approach is called *unsupervised learning* (learning without a teacher), where no matching outputs are specified for the input data. Therefore, for this type of learning, the output from the ANN cannot be directly checked and other criteria, such as network stability and convergence need to be applied.

The most commonly used method of supervised learning of multi-layered feed-forward ANN is the *error backpropagation* (BP). The method can utilize almost any activation function, provided that it is continuously differentiable [14]. Another advantage is that the ANN can have arbitrary number of layers with arbitrary number of neurons on each layer. The change of the synaptic weights is calculated according to the *Delta Rule* [14]:

$$\Delta w_{ij}(t) = \gamma \delta_i(t) y_j(t) . \quad (2)$$

The computation of error signals $\delta_i(t)$ for all neurons can be done using recursive formula, which specifies the error backpropagation in reverse direction from the outputs to the inputs of the ANN. If $\varphi'(in_i(t))$ is assumed as the derivative of the activation function, then [14]:

- the error signal of the neurons on *the output layer* is defined as:

$$\delta_i(t) = \varphi'(in_i(t))(ev_i(t) - y_i(t)) , \quad (3)$$

- the error signal of the neurons on *all the other layers* is defined as:

$$\delta_i(t) = \varphi'(in_i(t)) \sum_{h=1}^{N_0} (\delta_h(t) w_{hi}(t)) , \quad (4)$$

where N_0 is the number of neurons on the layer located to the right of the neuron y_i.

This method of learning is mostly suitable for control and modeling of technical systems, where it is possible to measure the system I/O parameters. The ANN can be used to identify the controlled system and create its model. It is also possible to use it in the role of neuro-controller in control of outputs of the system. In this case, the inverse model of the system is used.

2.1.2. Evolutionary and Population Based Optimization

Evolutionary algorithms (EA) represent a set of methods applicable to the optimization of parameters of various problems [19]. EAs are based on simulating the evolution of the population of individuals, where each individual represents particular solution to the optimization problem. These solutions are gradually changed by the EA using processes similar to those applied in the natural selection, such as reproduction and mutations. A general approach of the EA is shown on Fig. 7.

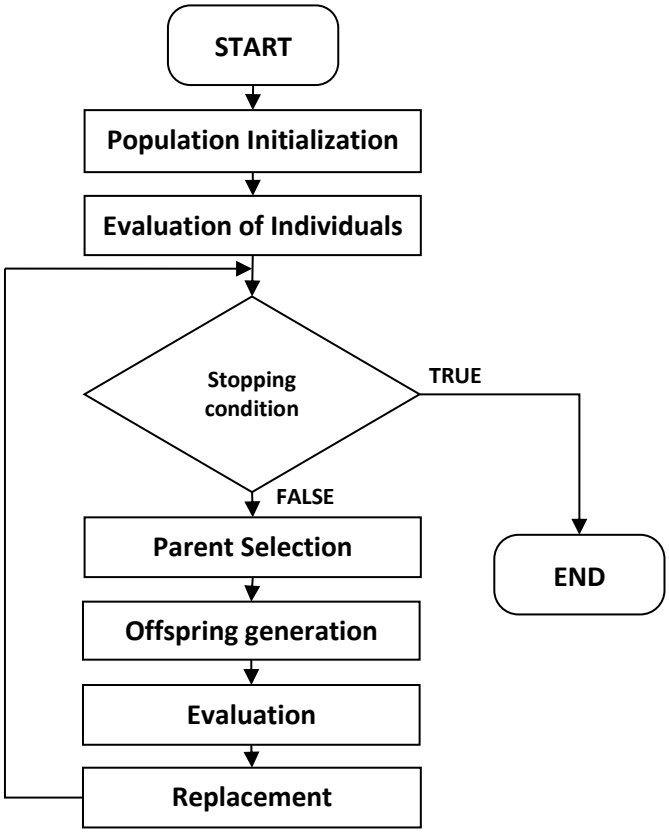


Fig. 7 General scheme of evolutionary algorithm.

We can also describe the EA as follows: If we assume that the task being handled consists of N components, then each solution (genotype of individual) represents one point in the N-dimensional space. EA works within (N+1) dimensional space where the additional dimension determines the suitability of the specific solution. The suitability (*fitness*) of the solution is considered a function of N variables, which can be represented as a hyper-plane in the (N+1) dimensional space (*fitness landscape* – see Fig. 8). The aim of the EA is to find the global extreme of the function (the best solution) and avoid local extremes [1].

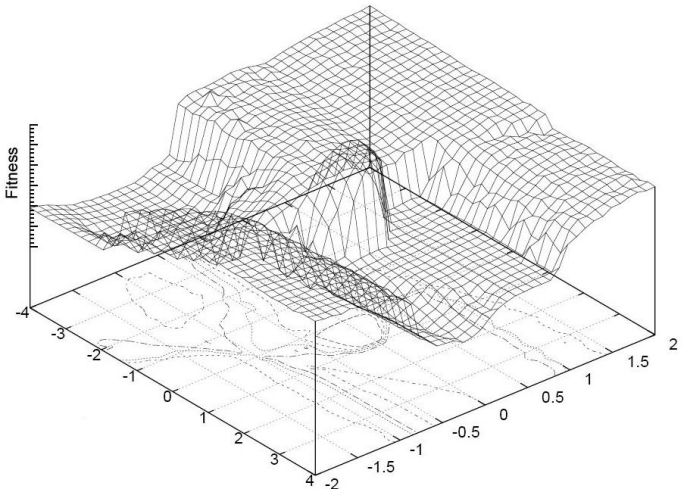


Fig. 8 Exemplary depiction of the fitness landscape [20].

The term EA covers a vast set of similar algorithms, strategies and approaches which are commonly inspired by the *theory of evolution*, e.g.:

- evolutionary strategies,
- evolutionary programming,
- genetic algorithms.

Population-based optimization algorithms such as *particle swarm optimization* (PSO) [21], as a modern alternative to EA, are very similar in the sense that they also work with the population of individuals representing the parameters of the problem addressed. However, changing the attributes of individuals is done in a different way than evolutionary. For example, in the case of PSO, the population is represented by particles moving around the fitness landscape according to a simple mathematical formula considering the position and speed of movement of the particle. Each particle moves in order to improve its current position, following the gradient of the landscape in the direction of its growth. In addition, it also considers the position of other particles if the fitness of their position is better. In this way, it is guaranteed to cover a wide range of available solutions with a high chance of finding global optima.

2.1.3. Fuzzy Systems

Fuzzy logic (FL) is very often used in various areas, including the control of nonlinear systems using *fuzzy inference systems* (FIS). The notion of *fuzzy* (unclear, blurry, vague) refers to the fact that some logical expressions cannot be clearly defined as true or false, but rather than partially true or partially false. Despite the fact that alternative approaches, including the *artificial neural networks* and *evolutionary algorithms*, can be applied to many problems with equal efficiency, fuzzy logic has an advantage that the solution is expressed in a way easily understood by a human, so that his experience can be used to improve the solution.

The central term in the area of FL is the *fuzzy set* (see Fig. 9). Compared to a conventional *crisp set*, the fuzzy set is different since it is not possible to clearly tell whether an element belongs to the set or not. In the fuzzy set, the whole range 0 to 1 is available to represent the element's *degree of membership* to the set, not just two values (0 or 1) as it is the case of crisp sets. Thus, fuzzy sets represent a generalization of a conventional set theory [22].

The fuzzy set can be defined as follows: If X is a set of objects (elements) x, then the fuzzy set A is a set of ordered pairs [22]:

$$A = \{(x, \mu_A(x)); x \in X\}, \tag{5}$$

where:

- $\mu_A : X \to M$ is the *membership function*,
- $\mu_A(x)$ is the *degree of membership* of the element x to the set A,
- X is the *universal set* comprising of all the elements.

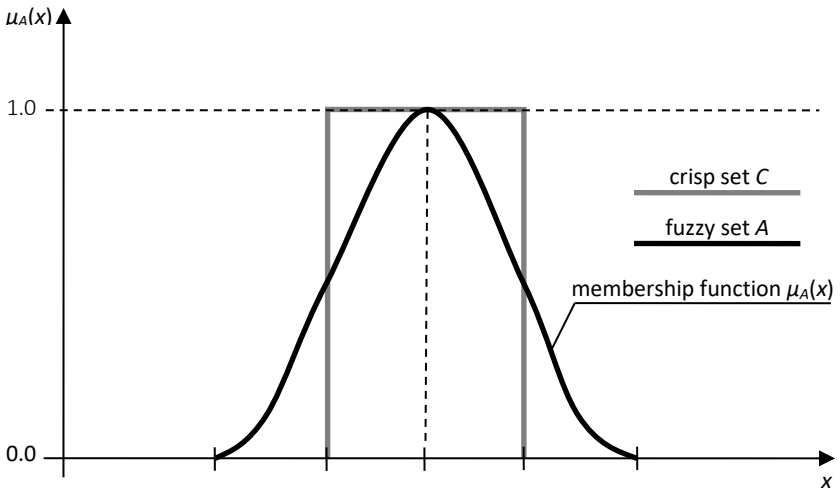


Fig. 9 Fuzzy set.

Fuzzy sets represent a basic building block for more complex logical structures with broader application. An example of such a structure is the *linguistic variable* (LV, see Fig. 10) [22], which is formally defined as the ordered tuple [22]:

$$LV = \{N, T(N), U, G, M\}, \tag{6}$$

where:

- *N* is a name of the linguistic variable (e.g. age),
- *T*(*N*) is a set of linguistic terms of the LV (e.g. young, adult, senior),
- *U* is the universal set,
- *G* are syntactic rules, which define the formation of names and terms of the LV,
- *M* are semantic rules, which assign appropriate fuzzy set to each linguistic term.

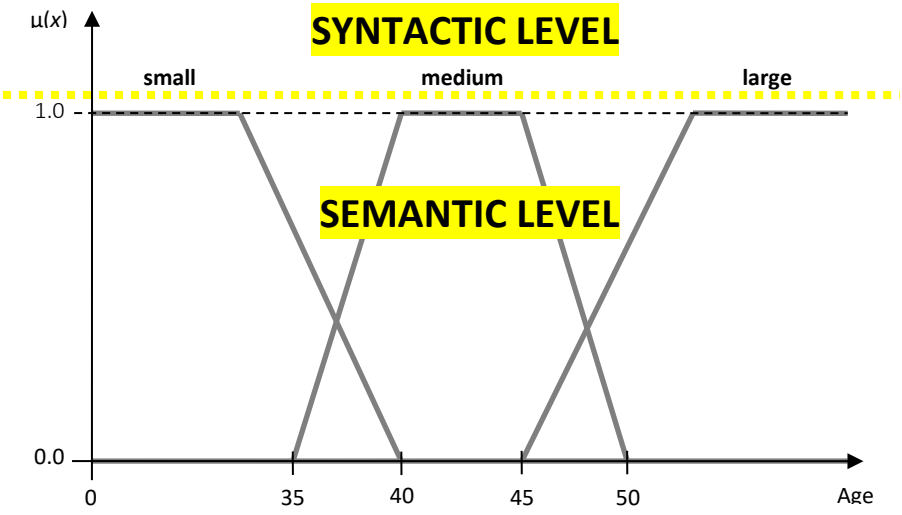


Fig. 10 Fuzzy linguistic variable.

A very useful application of linguistic variables is their application for creation of *fuzzy rules*. We can fuzzify the classic production rules by using linguistic variables (e.g. age, height, weight) and specific linguistic terms (small, medium, large). For example, we can create a rule such as:

- IF <*age* is *low*> & <*weight* is *high*> then <*life hazard* is *high*>.

Each fuzzy rule contains the *antecedent part* (the *premise*) and the *consequent part* (the *conclusion* or the *result*). The prerequisite part assigns input attributes (crisp numeric values) to different term (small, medium, large) of selected linguistic variables (age, height, weight of person) corresponding to the input attribute. The resulting part subsequently defines the linguistic variable for the output attribute and also determines the specific output linguistic term of this linguistic variable.

The *certainty of the rule* (or "firing of the rule") depends on the degree of membership of the input attribute to the fuzzy set of linguistic term used by the input linguistic variable. The problem arises if a rule has several assumptions in the antecedent part, each of which assigns input attributes to different linguistic terms with varying degrees of membership. In this case, the resulting degree of membership depends on the operators (the so-called T-norms and T-conorms) used to implement the logical joins *and* and *or* in a given rule. The most commonly used type of inference is *min-max*, where the join *and* is realized using the minimum operator and join *or* using the maximum operator [22].

The creation of a set of fuzzy rules is the basis for the design of *fuzzy inference systems* (FIS). The FIS is actually a set of production rules, the results of which is to be summed up or accumulated in a certain way. Visualization of the inference of fuzzy rules in FIS is displayed on Fig. 11.

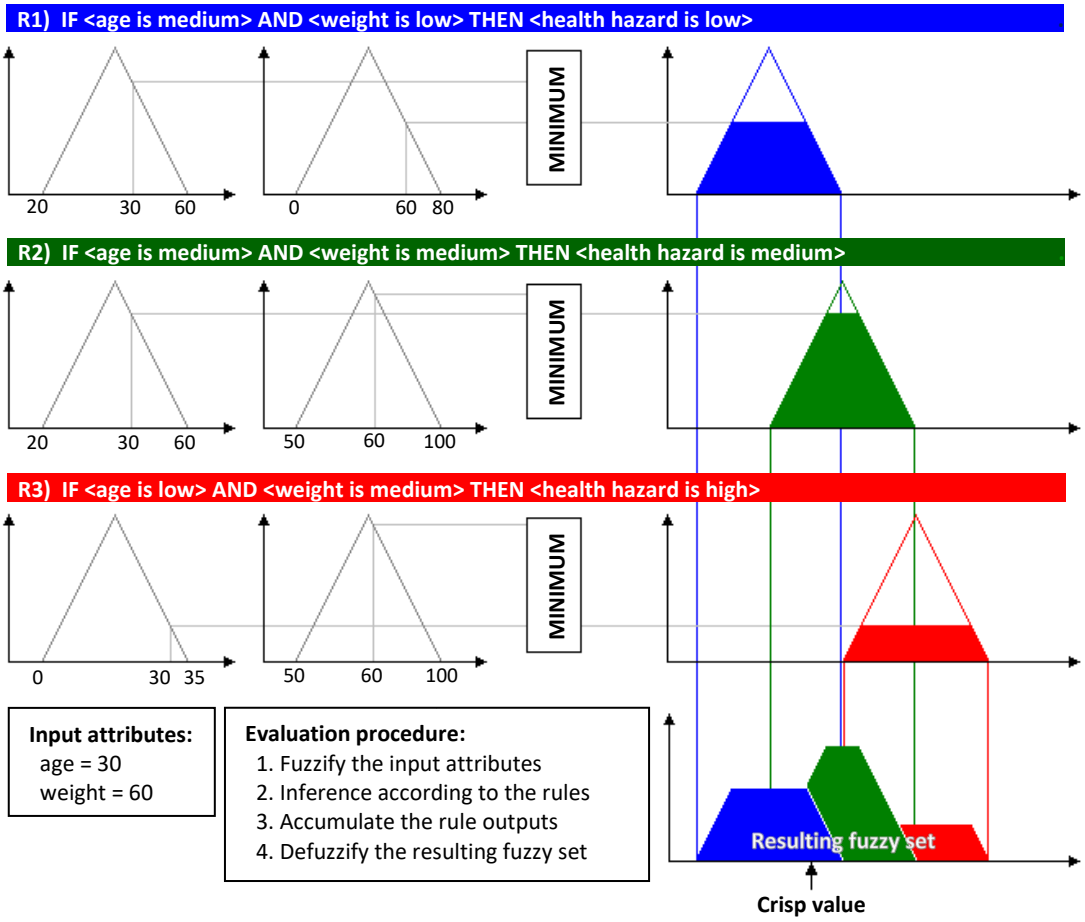


Fig. 11 Visualization of the inference of fuzzy rules in Mamdani FIS.

The inference process can be explained as follows: For each *input attribute* (or multiple attributes) the FIS forms a *fuzzy set* corresponding to the *output attribute*, while specific input attributes and output attributes are defined by *fuzzy rules* [22][23]. The determination of the output fuzzy set

depends on the rules themselves, in particular on the fuzzy sets of the linguistic terms in the consequent parts of the rules, but also from the certainty of the rules. The output fuzzy set determines the probability of an output attribute occurring across the entire universal set. The most likely (crisp) output value can be obtained by defuzzification of the resulting fuzzy set, using various approaches, e.g. the calculation of the center of gravity, maximum and others.

This example of the inference system is also referred to as the *Mamdani FIS* and is among the oldest types of FIS. It is also a basic prototype for the design of other derived types of FIS, for example the *Takagi-Sugeno-Kang FIS* (TSK) [24][25].

The structure and methods of inference of the TSK FIS are very similar to that of the Mamdani FIS. The difference is that the output of the rule is not a fuzzy set, but an analytical function. Therefore, the rules are defined as [22]:

- IF <attribute x_1 is high> & <attribute x_2 is low> THEN <output $u=f(x_1, x_2)$>.

In case of several rules, the final output u is calculated as a weighted sum [22]:

$$u = \frac{\sum_{i=1}^{m} \alpha_i u_i}{\sum_{i=1}^{m} \alpha_i}, \tag{7}$$

where m is the total number of rules, u_i is the output of i-th rule and α_i is its certainty.

The advantage of the TSK FIS is the direct calculation of the output attribute value. Since there is no need for costly defuzzification, it significantly increases the speed of computation, which is often very important. On the other hand, the exact definition of analytical functions defining the outputs of the rules can be difficult or sometimes even impossible.

In practice, the FIS can be used to deal with various tasks, e.g. development of expert systems, classifiers, predictors, nonlinear approximation models, system models, etc. For the purposes system control it is possible to use it in the design of fuzzy controllers. The block scheme of the Mamdani fuzzy controller is displayed on Fig. 12.

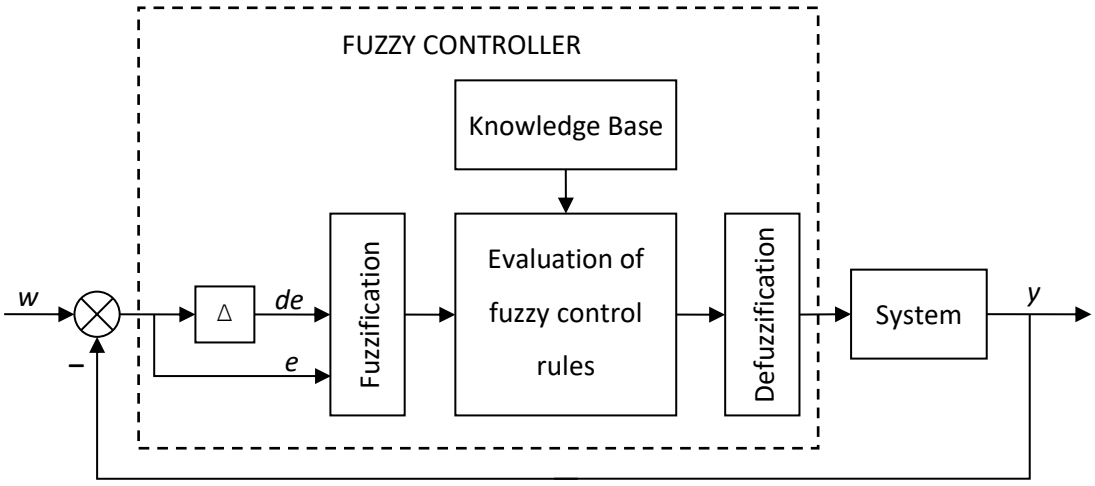


Fig. 12 Structure of fuzzy controller.

The special type of FIS are the *Fuzzy Cognitive Maps*, whose structure and method of inference is very similar to that of TSK. Their advantage is the simplification of the display of the rules in a graphic form that is clear and transparent also for a relatively large number of rules, as opposed to its explicit expression. This feature is particularly important when modelling systems with a large number of relations and interactions between individual elements. Fuzzy Cognitive Maps are explained in detail in Chapter 3.

2.2. Hybrid Control Methods

A *hybrid system* can be defined as a system that uses more than one method or technique to solve an abstract problem. Various methods can be used to create such systems [5][26]:

1) *Sequential hybrid approach* represents a simple serial involvement of two subsystems where the output of the first subsystem becomes the input of the second. This is the weakest form of hybridization.
2) A *complementary hybrid approach* uses one subsystem to optimize the parameters of the second subsystem. As with sequential approach, there is a possibility to distinguish both subsystems clearly. However, the hybridization level is higher in this case because the optimization subsystem is closely linked to the optimized subsystem.
3) The *reciprocal merger approach* (“embedding”) assumes mutual fusion of both subsystems. This approach can be considered as the highest form of hybridization because both subsystems are no longer uniquely identified.

In terms of implementation, hybrid systems are often proposed as decentralized systems. In view of the hierarchy of arrangement of individual elements, it is then possible to distinguish between two basic approaches [4]:

1) *Non-hierarchical system* (decomposable), where individual control subsystems are space-separated and ensure the control of different parts of the controlled system, without coordination at a hierarchically higher level.
2) *Hierarchically decentralized system*, where the central decision-making member determines which subsystem will be used for control during a given time or in a certain situation. When selecting a specific subsystem, either a simple switch condition [27] or more complex decision-making approaches can be used. Examples of such approaches are the methods of *multi-criterial decision-making* or means of *artificial intelligence*.

2.2.1. Hybrid Control of Technical Systems

In the context of *system control*, the concept of a *hybrid system* is used to designate systems that involve both continuous and discrete dynamics. In this sense, hybrid systems have a distinctive multi-level structure that includes [1]:

- A *lower level* (technological) representing a continuous system that represents physical reality;
- An *upper level* (decision/control), where a system with discrete events is used, which applies discrete computational methods for control, including the means of computational intelligence;
- A *contact interface* that allows the conversion of signals between the upper and lower levels using digital-analog converters and vice-versa.

A diagram of the such hybrid system is displayed on Fig. 13. The above-mentioned multilevel structure meets the requirements for modern digital control systems which expect the use of various mechanical and electronic components, including the interface with discrete computing systems [1][5].

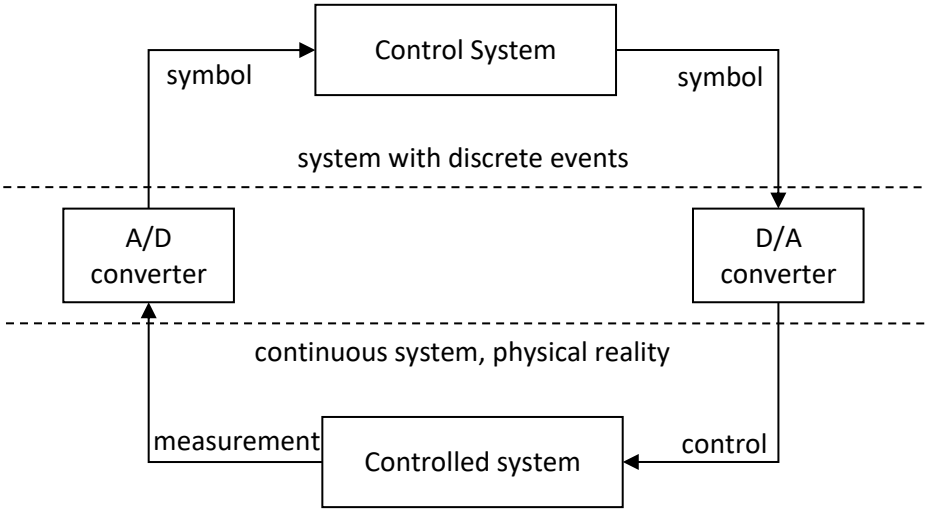


Fig. 13 Architecture of a hybrid control system [1].

In addition to the mentioned definition of the hybrid system, this concept can also be used exclusively for the examination of combined control systems within the upper level. In the next section we will discuss the hybridization of control methods based on the means of computational intelligence that can be used at this level.

2.2.2. Hybrid Intelligent Systems

Hybrid intelligent systems represent a group of algorithms that combine different approaches. They can represent the integration of classical control methods with modern approaches using elements of AI. Their mutual integration is intended to reconcile the benefits of individual methods. An overview of the advantages and disadvantages of CI as compared to the methods of symbolic UI and algorithms based on conventional control theory is given in Tab. 1 [28].

Tab. 1 Qualitative comparison of AI methods [28].

	FS	ANN	EA	CT	SAI
Mathematical model	2	4	4	1	3
Learning capability	4	1	2	4	4
Knowledge representation	1	4	3	3	1
Expert knowledge	1	4	4	3	1
Nonlinearity	1	1	1	4	3
Optimization ability	4	2	1	3	4
Fault tolerance	1	1	1	4	4
Uncertainty tolerance	1	1	1	4	4
Real-time operation	1	2	3	1	4

Legend: FS – fuzzy systems, ANN – artificial neural networks, EA – evolutionary algorithms, CT – conventional control theory, SAI – symbolic artificial intelligence. Lower score is better.

Further, we focus on the possibility of hybridization of individual means of computational intelligence, i.e. artificial neural networks, fuzzy systems and evolutionary algorithms (see Fig. 14). In terms of system control, the possibility of combining ANN and FIS seems the most appropriate, as both methods integrate inference systems capable of transforming control signals. By combining these approaches, we create hybrid *neuro-fuzzy systems*.

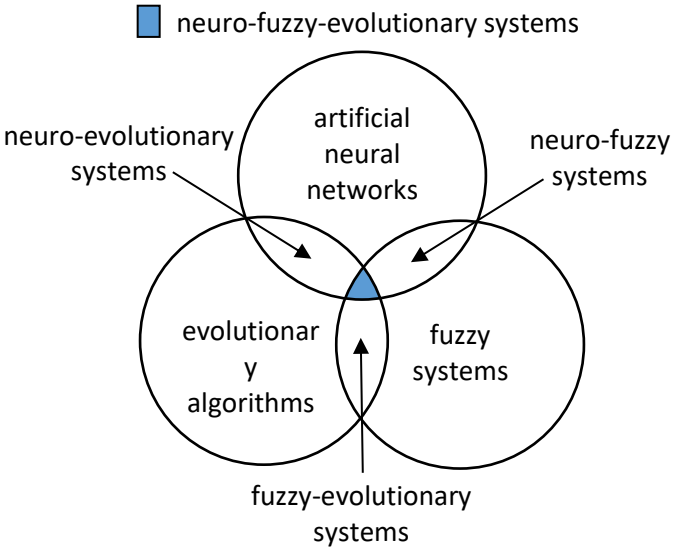


Fig. 14 Hybridization of computational intelligence [1].

The EA as an optimization tool are very often used as an alternative to setting up parameters of both ANN and FIS. With FIS, the EA can be used to automatically generate rules into the knowledge base, which is particularly useful if the expert knowledge needed to solve the problem is not available. In the case of its use for the adaptation of the ANN parameters, the EA may be used as a substitution for conventional supervised learning using error backpropagation.

2.2.2.1. Neuro-Fuzzy Systems

The universal approximation theorem (see chapter 2.1.1) says that both ANN and FIS are able to approximate any function. However, it is not always evident what number of rules or neurons is sufficient to address a particular approximation problem. The determination of an appropriate number (of either rules or neurons) may be simplified by employing the combination of ANN and FIS into a suitable hybrid structure. The FIS and ANN hybridization can be implemented in several ways. In particular, the following options are available [1]:

1) If the fuzzy logic is implemented into neurons, the so-called ***fuzzy neurons*** are created. This modification allows the ANN to process *uncertainty*. The knowledge stored in such fuzzified ANN can be considered more abstract, as the network is not limited to work with crisp values.
2) There is also an opposite approach, i.e. the ***implementation of the FIS using ANN***. If a fuzzy controller is implemented in such a way, it has the ability to learn and automatically construct its knowledge base. Hence, the disadvantages of FIS are diminished by the use of ANN. The most familiar architecture of this type are *ANFIS*, *FALCON* and *NARA* (see Fig. 15).
3) ANN can be used as a mean to ***generate a rule base*** for a fuzzy controller (if it is possible to apply supervised learning). An example of such usage is the adaptive fuzzy controller using *adaptive fuzzy associative memory* (AFAM).
4) Another possibility for interaction is to use the fuzzy controller in order to ***adapt the learning parameters of ANN*** according to the current needs during the learning process. The appropriate change in learning parameters at different learning stages allows for faster convergence of neural network parameters to optimal values.
5) In addition, it is possible to create ***hybrid systems*** in which both ANN and FIS are separate and perform separate tasks. They may be combined in parallel when their outputs are combined or mutualy modified. They may also be serially connected, when the output of one of the systems becomes the input of other.

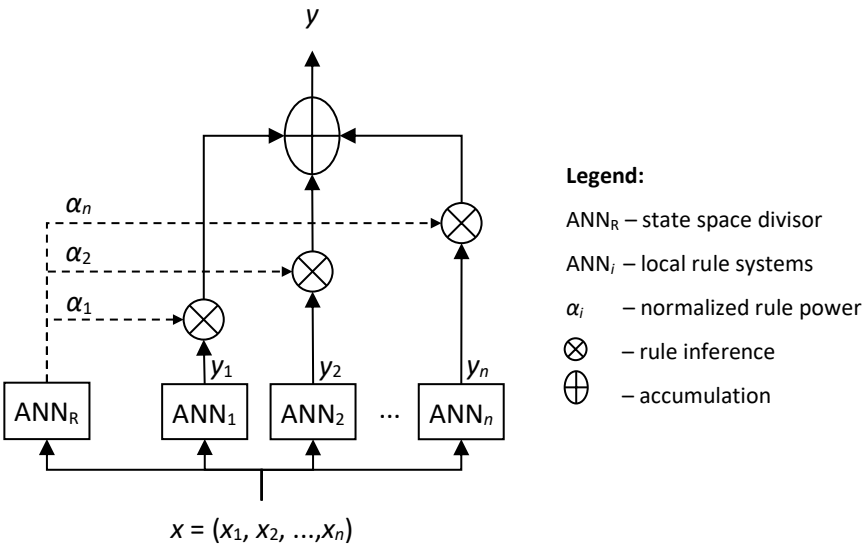


Fig. 15 NARA – Neural-network designed on Approximate Reasoning Architecture.

2.2.2.2. Neuro-Evolutionary Systems

The use of EA in learning of ANN, or neuro-evolution [29], is an alternative to conventional machine learning algorithms based on supervised learning, which inevitably need training data defining the corresponding inputs and outputs of the approximated function. Because EA does not need such training data, it can be used in a much wider range of applications. The only prerequisite is the ability to define a criterial function determining the suitability of individuals representing the parameters of the ANN. An example of a possible application is a game where the result (win or loss) can be easily determined without the need for training samples of suitable gaming strategies.

There is a large number of various neuro-evolutionary algorithms. From the perspective of adaptation of the structure of ANN they may be divided as follows:

- *Evolution of ANN with a fixed topology*, where the structure of the ANN is immutable and only the synaptic weights between neurons are being changed.
- *Evolution of the topology and the weights of the ANN* (TWEANN – "Topology & Weight Evolving Artificial Neural Network algorithm"), which also makes it possible to change the structure of the ANN.

If the method of encoding the parameters of ANN (phenotype) into the genotype of an individual is considered, there are two fundamental approaches [30]:

- *Direct encoding* of the parameters of the ANN, which assumes direct mapping of the phenotype into the genotype and vice versa. Thus, all neurons and weights in the ANN are in the genotype of an individual specified explicitly.
- *Indirect encoding* of the parameters of the ANN, in which the genotype of an individual does not explicitly specify the structure of the ANN, but only indirectly defines the method of its construction. This approach is especially advantageous when using TWEANN algorithms to design recurrent neural networks, where the individual structure of the ANN can be repeated. Another advantage is the compression of the phenotype to a smaller genotype, thus decreasing the size of the search space and ultimately increasing the speed of EA's convergence to the optimal values.

2.2.2.3. Fuzzy-Evolutionary Systems

When combined with fuzzy systems, the EA is an alternative to the traditional creation of the rule base using expert knowledge. For this purpose, it is possible to use the EA, where each individual of the population represents one fuzzy rule. The new rules are obtained by the evolution of the initial base of randomly generated rules. After a certain number of generations, the best individual from the population is included in the final rule base.

An example of such FIS and EA combination is is the Feature Selective Linguistic Classifier (FeSLIC) [31], which is actually a genetic fuzzy rule-based classification system created using iterative rule learning (IRL).

The IRL method resolves the issue of rule base creation by gradually adding one rule after another by re-starting a genetic algorithm. A set of sample training data is used to determine the fitness of the rules. Whenever a new rule is produced, training samples that are sufficiently covered by it are removed from the training set, so the newly created rules can focus on the correct classification of other samples. However, this method can lead to conflicting rules, since the rules created at later stages of algorithm runtime do not take account of previously removed training samples [32].

The solution to this problem is to use the AdaBoost algorithm [32], which is suitable for creating rule base using IRL [33]. AdaBoost (see Fig. 16) assigns to each training sample the appropriate weight representing its relative importance in further learning. Whenever a new rule is produced, weights in those samples which have been classified correctly are reduced, allowing for new rules to concentrate on improperly classified samples. However, since none of the samples is completely removed from the training set, the newly created rules are aware of the previously created ones, which limits the creation of conflicting rules.

The advantage of evolutionary generated rule systems in comparison to the expert proposal is the automation and overall acceleration of the FIS design process. A disadvantage is often a lower comprehensibility and a large number of rules generated. However, using additional optimization methods [31], these shortcomings can be reduced to a certain extent, for example by eliminating redundant rules or by filtering out redundant clauses in the prerequisite parts of the rules.

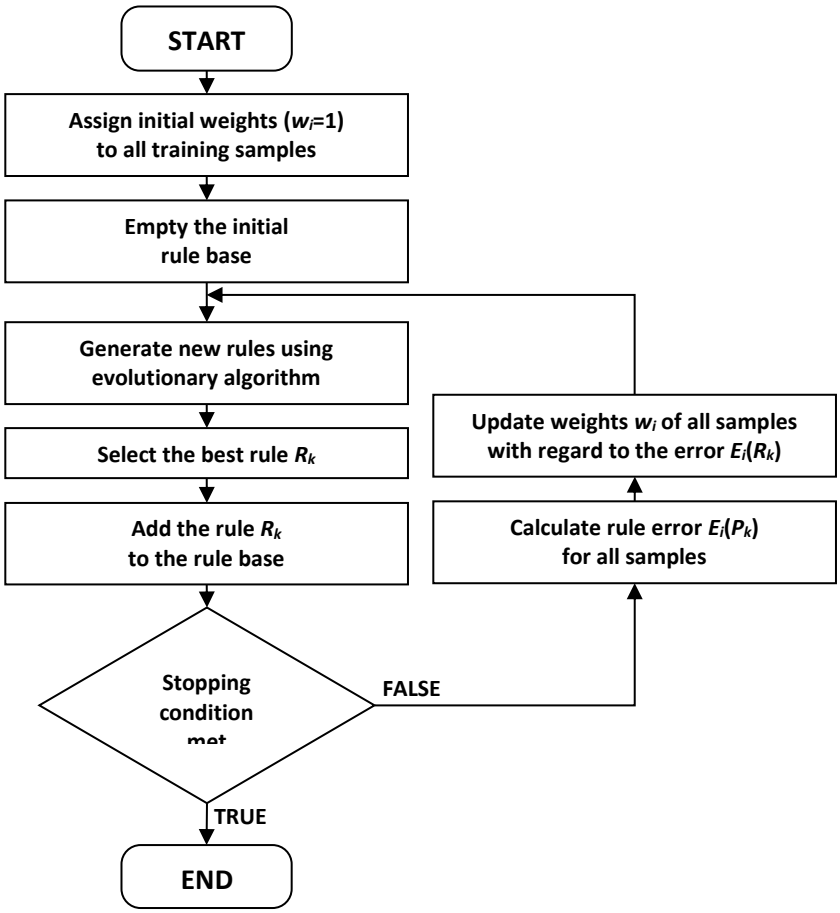


Fig. 16 Iterative rule generation using AdaBoost algorithm.

2.3. Utilization of AI for Situational Control and Modeling

One of the objectives of situational control is the control of the system in every operational state. In addition to normal operating conditions, it is also necessary to provide control mechanisms during system faults. In order to determine non-standard conditions during operational control, the ability to properly diagnose controlled ZS is crucial. During planning phase, before the operational control itself, it is important to create suitable situational models of the complex system. The adaptive algorithms and intelligent methods can be used in all of these areas [5].

Evolutionary approaches can be used to adapt the parameters of control algorithms. For example, EA can be used in order to achieve a purposeful change of the system structure. The consequences of this change can be evaluated within the scope of the current situation. If the change is beneficial, it can be incorporated into the next generation of applicable control structures [5].

The situational control can be implemented by using several cooperating subsystems involved in a control of the entire complex system in a given situation. Cooperative control can be used as well. It can be applied at several levels with varying degrees of cooperation and communication between individual subsystems [34]. For the implementation, it is viable to use ANN, FIS or hybrid neuro-fuzzy systems such as *fuzzy cognitive maps* (further explained in Chapter 3).

The framework of cooperation between subsystems can be established using supervisory control element – the *formatter*. The formatter manages the function and form of the complex system using situational control by assigning command to individual subsystems depending on the current situation [34]. The diagram in Fig. 17 shows the situational control model implemented using the formatter. The model presumes utilization of means of computational intelligence in order to implement partial control functions.

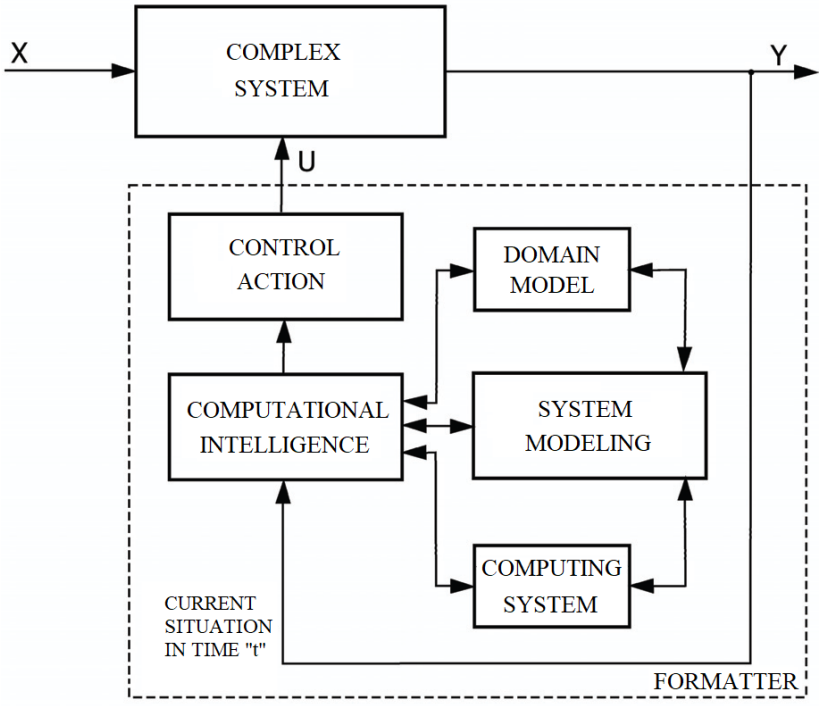


Fig. 17 Situational control using means of computational intelligence [1].

Formatter is generally a control system that controls the function and form (external appearance and internal structure) of a complex system [35]. Its general structure is shown in Fig. 18. It can be implemented by one computer or by decentralized network of multiple computers. It is typically

part of a feedback loop with a controlled complex system, where feedback is obtained using the information about the controlled system, namely the vectors of input variables X, output variables Y and internal state variables Z. The control vector R from the command system, which determines the control objectives, is also part of the inputs of the formatter. Individual input vectors are analyzed by appropriate analyzers (ANX, ANY, ANR, ANZ). Analyzers transform and reduce the received information (for example, they can perform situational classification). The results of their analysis are passed to the supervisory controller which determines the control actions suitable for the current situation in a form of the action vector U [1].

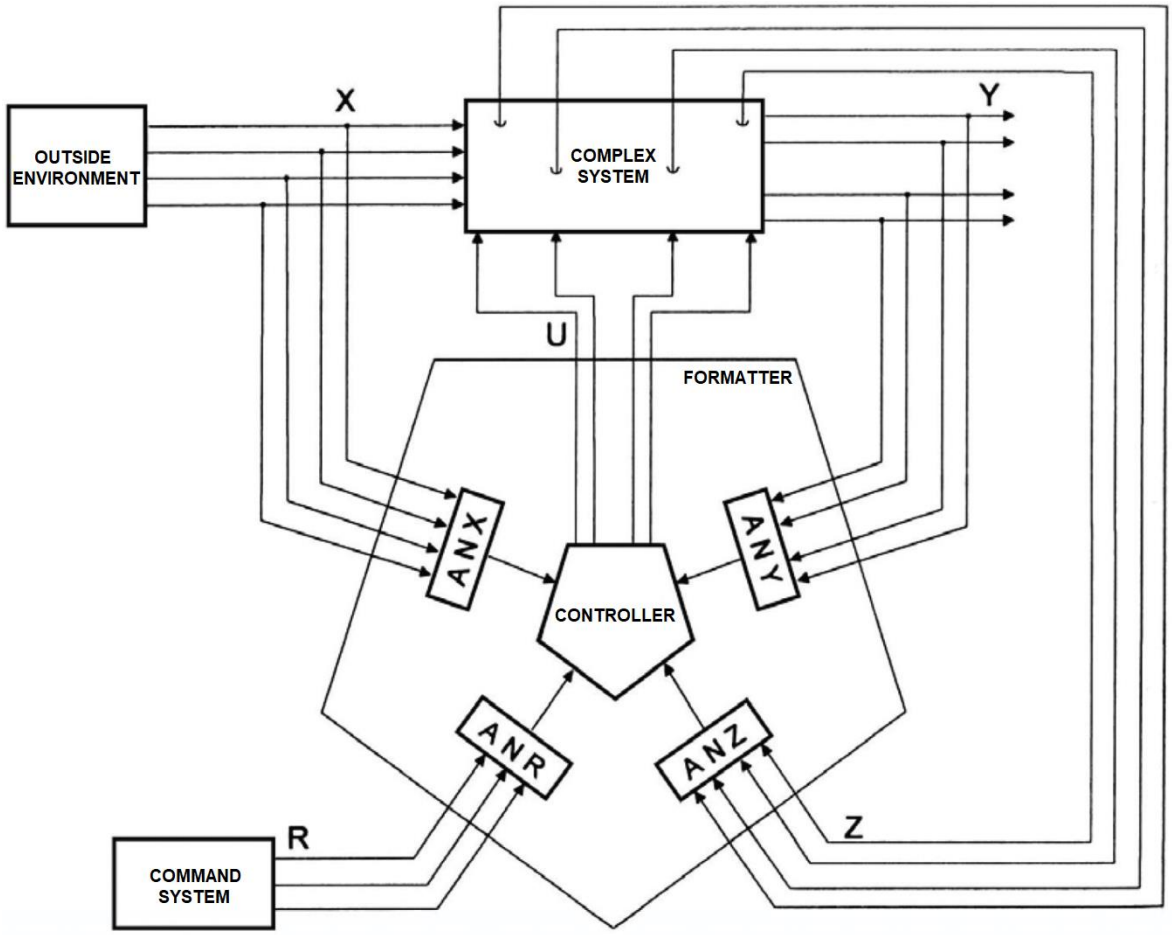


Fig. 18 Formatter control [1].

The operation of the formatter (including the individual analyzers and the controller) consists of a *decision* and *control* phase, each divided into a *selection* part and *action* part. Accordingly, the control process consists of four stages [6]:

1) The processed *situation* is analyzed in the *selection part* of the *decision phase*. According to the analysis, it is assigned to one of the N situational classes. Each situational class has a set of control algorithms available.

2) During the *action part* of the *decision phase*, a set of algorithms best suited for processing a current situation is activated.
3) During the *selection part* of the *control phase*, the selected algorithms are adapted (parameterized) in order to address the situation accordingly.
4) Realization of the control takes place in the *action part* of the *control phase*.

This model of *situational control* in the form of a *formatter control* is suitable to be implemented with application of intelligent methods [5]. According to the model, it is possible to develop a specific implementation of the situational control system, including the definition of the individual action and decision-making elements implemented by the means of AI [5]. With regard to the properties of the individual means of computational intelligence, it is obvious that ANN and FIS are applicable in particular to the role of analyzers of individual state variables, for instance in situational classification. Their use is also possible for implementation of specific control algorithms. On the other hand, the EA can be used to parameterize these control algorithms.

3. Fuzzy Cognitive Maps

"Few people are capable of expressing with equanimity opinions which differ from the prejudices of their social environment. Most people are incapable of forming such opinions."

Albert Einstein

Fuzzy cognitive maps (FCM) are a widely used inference tool for modeling qualitative and quantitative complex relationships across a wide range of different technical and non-technical systems in a simple and comprehensible way. In the last decade, FCMs has played an important role in applications in many areas of scientific research such as social and political science, engineering, information technology, robotics, expert systems, medicine, education, prediction, the environment and others. An extensive overview study [36] of leading personalities in FCM research, namely prof. E. Papageorgiou and prof. J. Salmeron, includes the following areas as typical FCM applications:

- *management* - prediction, interpretation, monitoring,
- *business* - planning, management, decision making, inference,
- *medicine* - decision support, modeling, prediction, classification,
- *robotics* - navigation, learning, prediction,
- *environment* - representation, knowledge derivation and thinking, policy making,
- *information technologies* - modeling, analysis.

The same study analyzes the reasons for such a wide applicability of FCM in different areas. The main assumptions that support the use of FCM are in particular [36]:

- simplicity of design and determination of parameters,
- flexibility of representation (it is easy to add / remove new concepts)
- ease of use, clarity and transparency for non-technical experts,
- low computational complexity,
- managing dynamic effects thanks to the feedback structure.

For these reasons, simple and intuitive creation of system models is also possible for non-technical workers and general public. For comparison, the other methods, such as *Bayesian networks* or *Petri nets,* do not provide an obvious way to create similar system models, especially to an uninvolved person without prior experience [37][38]. On the other hand, FCM allows the simultaneous

involvement of multiple experts in the system modeling process, since it is easy to combine several individual FCMs, each created by individual expert, into more complicated structures. There is no such option for Petri nets, as it is not generally established how various Petri nets, describing the same system, can be associated and connected together [37][38].

In the following subchapters, we will introduce the basic definitions of FCM and look at the way of their inference and computations. We also outline the basic deficiencies of traditional FCMs which may limit their application. In the end of the chapter, we will introduce the automated setting of FCM parameters using the various available machine learning methods.

3.1. Cognitive Maps

Generally, the *cognitive map* (CM) [39] is an *oriented graph* (see Fig. 19), where the nodes represent concepts (basic terms) and the edges determine their mutual causal relationships. Concepts typically determine states and edges are actions or functions that transform the state of one node into the state of another node. The effects of relationships between concepts are usually defined by trivalent crisp values: -1, 0 and 1 [40].

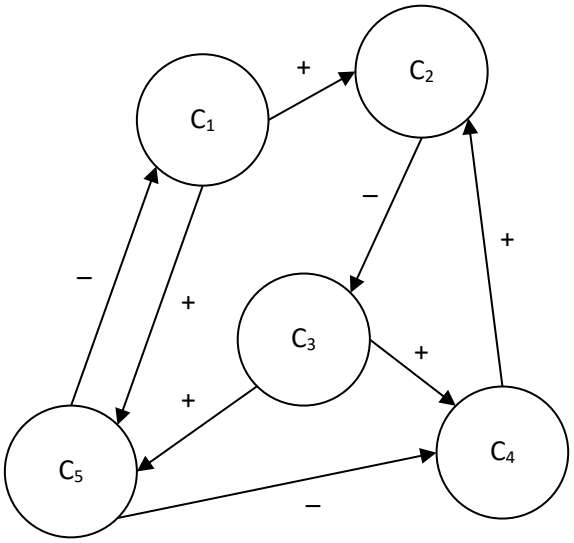


Fig. 19 Simple cognitive map.

Due to their cyclic structure, CMs are able to describe complex dynamic systems. It is possible to examine limiting system cycles [41], collisions, etc. Additionally, CM's greatest advantage is their easily comprehensible representation of knowledge that can be visualized graphically [41]. An illustrative example of the use of CM is displayed in Fig. 20, which models the impact on the political situation in Lebanon in the early 1980s [42]. Since the design of such a map is very simple and intuitive, it is clear that CM represents a suitable tool for rapid prototyping of system models.

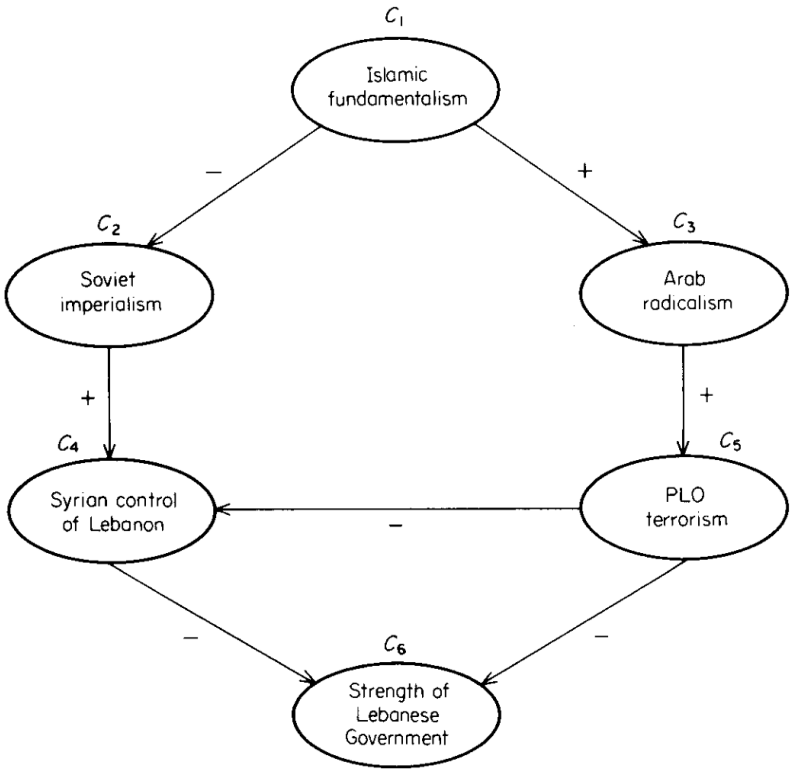


Fig. 20 Cognitive map of political situation in Lebanon constructed on basis of the newspaper article by Henry A. Kissing named "Starting Out in the Direction of Middle East Peace" (Los Angeles Times, 1982) [42].

The disadvantage of CMs which utilizes sharp values is their limited ability to model more complex relations between concepts. For this reason, the basic concept of CM was extended, resulting in fuzzy cognitive maps.

3.2. Fuzzy Cognitive Maps

Fuzzy cognitive maps (FCM) as an extension of classical CMs were designed by Kosko [42] in 1986. FCMs are (as well as classical CM) graphs that model the system and its behavior using concepts and their mutual relationships that can be positive, negative or none [40][43][44]. However, the FCM (compared to the classic CM) also considers the degree of influence (or weight) of these relationships expressed by the respective value from the continuous interval [-1,1] (see Fig. 21).

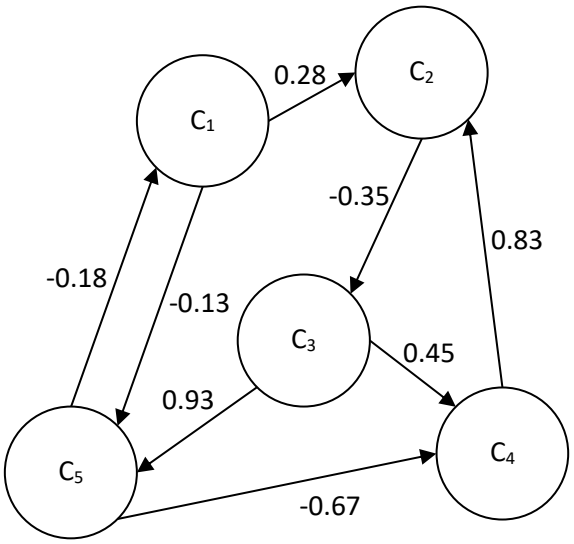


Fig. 21 Fuzzy cognitive map.

In addition to the representation of FCM in the form of a graph, it is also possible to use the representation of relations between concepts using the *connection matrix* (or *adjacency matrix*) *W*. Matrix representation is often useful for computations using spreadsheet processors (such as Excel) as well as other programs that support matrix computations (MATLAB, Scientific Python etc.) For FCM on Fig. 21, the corresponding matrix *W* is determined as follows:

$$W = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & C_4 & C_5 \end{array} \\ \begin{pmatrix} 0 & 0{,}28 & 0 & 0 & -0{,}13 \\ 0 & 0 & -0{,}35 & 0 & 0 \\ 0 & 0 & 0 & 0{,}45 & 0{,}93 \\ 0 & 0{,}83 & 0 & 0 & 0 \\ -0{,}18 & 0 & 0 & -0{,}67 & 0 \end{pmatrix} \end{array} \tag{8}$$

Note that the structure and possible representations of the FCM (i.e. *graph* or *matrix*) fully correspond to the description requirements of complex systems as outlined in chapter 1.

3.3. Formal Definition of FCM

In order to facilitate text readability, we have mentioned only very loose definitions of CM and FCM. For completeness, however, a more formal mathematical definition is possible. There are a several varying FCM definitions, but the most commonly used is the definition of Chen [45], where FCM is defined as a quadruple [40]:

$$FCM = (C, W, \alpha, \beta) \tag{9}$$

where:

- $C = \{C_1, C_2, ..., C_n\}$ is a final set of cognitive units (*concepts*),
- $W = \{w_{11}, w_{12}, ..., w_{nm}\}$ is a final set of oriented connections between concepts,
- $\alpha \rightarrow$ <0, 1> maps the real value to the degree of membership in the concept (*fuzzification*),
- $\beta \rightarrow$ <-1, 1> is similar to α, but determines mapping of the edges (or *weights*).

The concept C_i (i = 1, 2, ..., n) represents the state, procedure, event or system variable [44] and its value is from the interval [0, 1] or (according to the implementation) from the interval [-1.1]. The oriented connection between the concepts C_i and C_j determines the causality between these concepts and is represented by the weight w_{ij}, which is determined by the fuzzy value from the interval [-1, 1]. For each weigh w_{ij} can distinguish these three options [44]:

- the weight has a *positive value* ($w_{ij} > 0$), so the increase in the value of the C_i concept leads to an increase in the value of the concept C_j.
- the weight has a *negative value* ($w_{ij} < 0$), so the increase (decrease) in the value of the concept C_i leads to a decrease (increase) in the value of the concept C_j.
- the weight has a *zero value* ($w_{ij} = 0$), meaning there is no relationship between the values of the C_i and C_j concepts.

3.4. Concept Activation and Inference in FCM

A general rule to determine the *activation value* of the concept C_j in every simulation step is to determine the influence of other preceding concepts C_i (where i = 1 ... m), which are connected to the concept C_j (see Fig. 22).

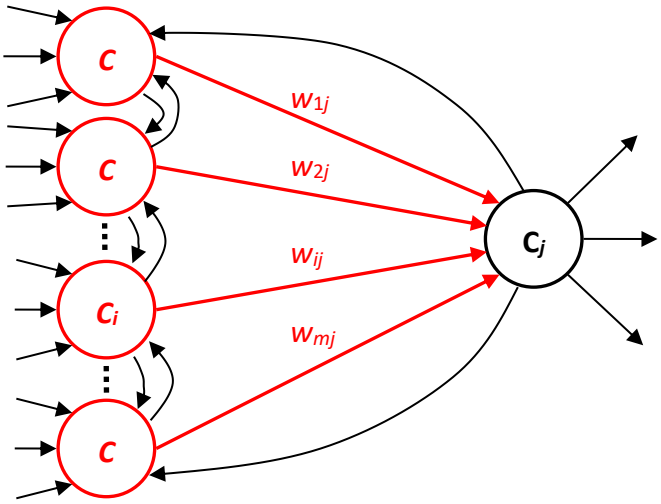


Fig. 22 Calculation of the activation of concept C_j depends on the values of preceding concepts C_i.

If $A_j(t)$ is the activation value of the concept C_j in time t, then the activation value $A_j(t+1)$ of the concept C_j in time $t+1$ is determined using the activation values $A_i(t)$ of the preceding concepts C_i in time t as follows [44]:

$$A_j(t+1) = p\left(A_j(t) + \sum_{i=1;i\neq j}^{n} w_{ij} A_i(t) \right) \tag{10}$$

where w_{ij} are the weights on the edges between preceding concepts *Ci* and the following concept *Cj*. The thresholding function p limits the result to the values from the interval [-1,1] [44]. A behavior of FCM depends greatly on the used thresholding function [43]. The most commonly used functions $p(x)$ are [43]:

- bivalent:

$$p(x) = \begin{cases} 0; & x \leq 0 \\ 1; & x \geq 0 \end{cases}, \tag{11}$$

- trivalent:

$$p(x) = \begin{cases} -1; & x \leq -0{,}5 \\ 0; & -0{,}5 \leq x \leq 0{,}5 \\ +1; & x \geq 0{,}5 \end{cases}, \tag{12}$$

- logistic (sigmoid):

$$p(x) = \frac{1}{1+e^{-mx}} \tag{13}$$

The sigmoid threshold functions or other continuous functions whose result is in the range [0,1] are the most commonly used. This interval is particularly important when transforming real values (e.g. sensory signals, physical quantities, statistical values, etc.) into degrees of membership in a given concept during fuzzification.

4. System Control and Modeling using Fuzzy Cognitive Maps

"You may not control all the events that happen to you,
but you can decide not to be reduced by them."
Maya Angelou

The system model is typically constructed on the basis of expert knowledge or using data measured during system operation. The model may be later used to solve tasks such as prediction, diagnostics, interpretation, monitoring, control, classification, etc. When artificial intelligence means such as FCM are used to create a model, this process is often called model learning. However, not all means of CI are suitable to solve any control or modeling situation. Therefore, in this chapter, we focus on description of basic problems and tasks solvable by FCM and further discuss possible hybrid systems with ANN or EA, which could enable FCM utilization in other areas.

4.1. Basic Problems Solvable with FCM

According to [46], the FCM are suitable to deal with two main problem types, namely:

1) the creation of ***regression model*** of the system,
2) the ***encoding of the attractor*** leading to specific goal state of the system.

The first type (*regression*) is the most frequent task for which FCM is used. Its goal is to set the parameters of FCM (which are usually weights on the links between concepts) to create a corresponding real-system model. Conventional FCMs are most commonly used for modeling closed systems without external inputs (see Fig. 23). However, after some modification of the structure and computation process, it is also possible to model open systems [47]. The most important change is the introduction of special input concepts that represent faults, noise and other influences from outside environment including control signals [47].

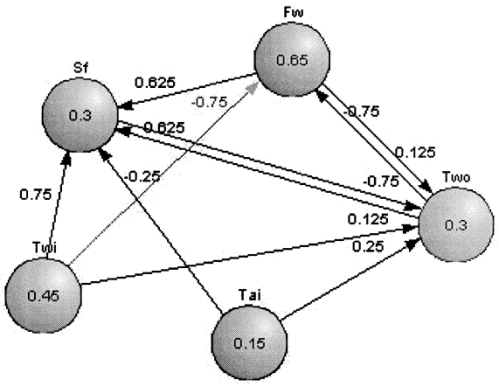


Fig. 23 Model of heat exchanger realized using FCM [48].

The second type (*attractor definition* [41], see Fig. 24) deals with problems of stability, convergence, and ultimately, control of FCM. The goal of this task is to encode the convergence capability from any initial state into the desired target state. The target state of the FCM is determined by the specific values of the selected set of *controlled concepts*. In other words, we try to find such values of FCM parameters (usually weights between concepts) in which the system represented by the given FCM gets from the various initial conditions into the desired final stable state or stable limit cycle [41]. In this case, we do not necessarily have to set all the FCM parameters because the model of the *controlled system* can be pre-defined. The aim is to determine only the parameters (weights) of the selected *controller concepts*.

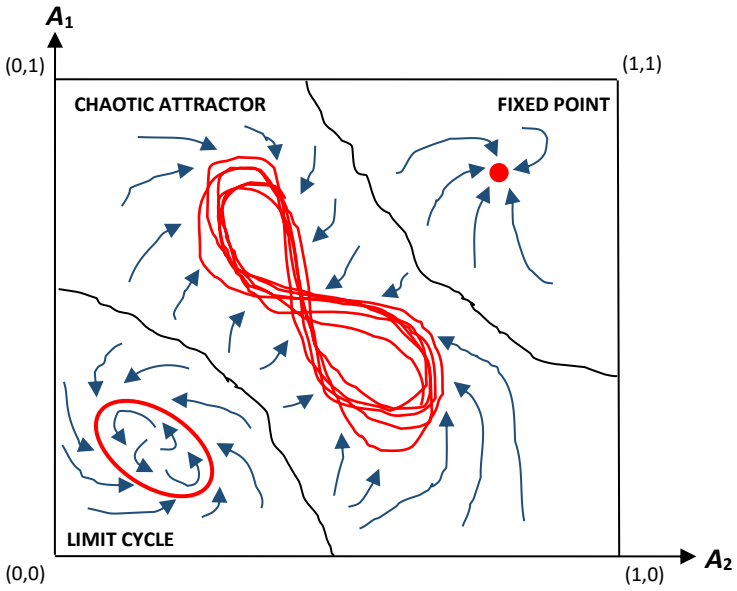


Fig. 24 Visualization of three various attractors of the same FCM with two concepts selected.

With regard to the mentioned basic tasks solvable with FCM, it is clearly visible that FCMs can be used to tackle problems in the field of system control and modeling. Basically, tasks of the *first class* are solved in relation with *system modeling*, while tasks of the *second class* are applicable to *system control*.

4.2. Modifications of FCM for Control of Complex Systems

The use of intelligent methods for situational control of complex systems [53] is possible for:

- the *situational classification* of the current state into the appropriate situational class,
- the *design of intelligent control* methods for some operating states,
- the *design of models* used to diagnose the system condition.

FCM as a mean of CI are available in all of these areas. However, certain limitations (see Chapter 4.2.1) of FCMs have to be considered. Relationships between system elements in the real world are neither linear nor monotonic, as is the case with models implemented using simple FCMs [36]. Real systems usually have non-linear dynamics, often higher than second-order. Additionally, the FCM dynamics is only the first-order where the next state depends only on the previous one.

The next sub-chapters present the ***first contribution*** of this dissertation, which is *"the modification of the basic concept of FCM in order to allow the modeling and control of complex systems."* Most of its content is taken from and was published in the previous publication [55] of the author.

4.2.1. Disadvantages of Simple FCM

There are several limitations of conventional FCMs applied to the modeling of complex systems. These can be summarized as follows [36][38]:

- Relation weights are just linear (see Fig. 25).
- Models lack time delay in the interactions between nodes.
- The FCM cannot represent logical operators (AND, OR, NOT, and XOR) between parallel incoming nodes.
- It cannot model multi-meaning (grey) environments.
- Multi-state (quantum) of the concepts is not supported.
- No more than one relation between nodes is allowed.
- Many real-world causal relations are neither symmetric nor monotonic as within the FCM model.
- The FCM dynamic is the first order, where the next state depends just on the previous one.

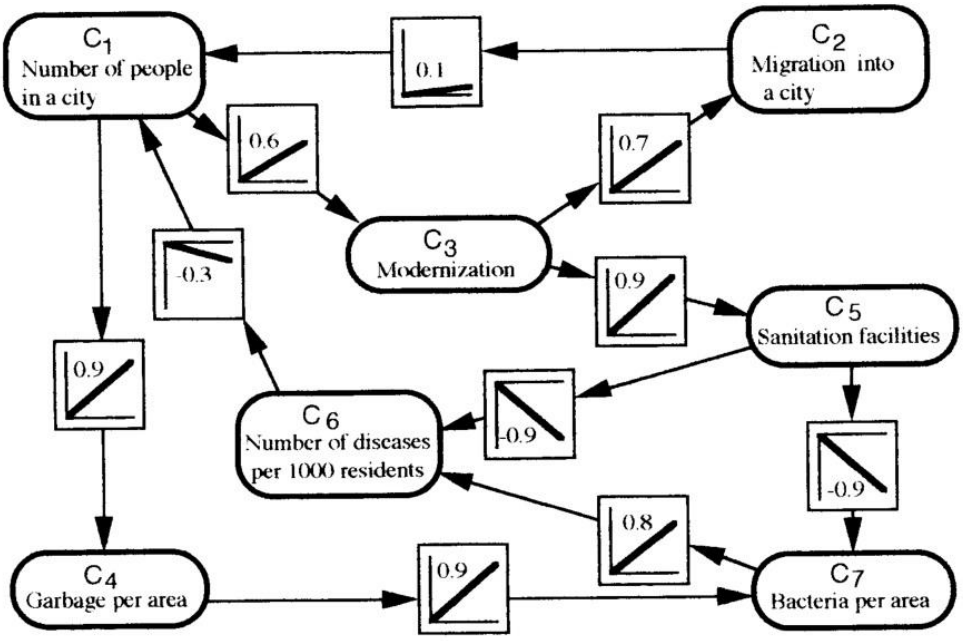


Fig. 25 A visualization of the concept relations within the FCM model of a city. It is clearly visible, that the relations can be represented only as linear functions [54].

4.2.2. Established FCM Extensions

If we take the shortcomings of the FCM from the previous section into account, it is clear that basic FCM may not be sufficient to model complex systems. As a result, it is necessary to suggest some modifications to the basic structure of FMC. In order to overcome problems of the basic FMC, such as a linearity, first-order dynamics, a lack of time delay and an inability to represent logical operators, the following already established extensions can be taken into consideration (see Fig. 26) [54]:

- weights extended to nonlinear functions,
- conditional weights,
- time-delayed weights.

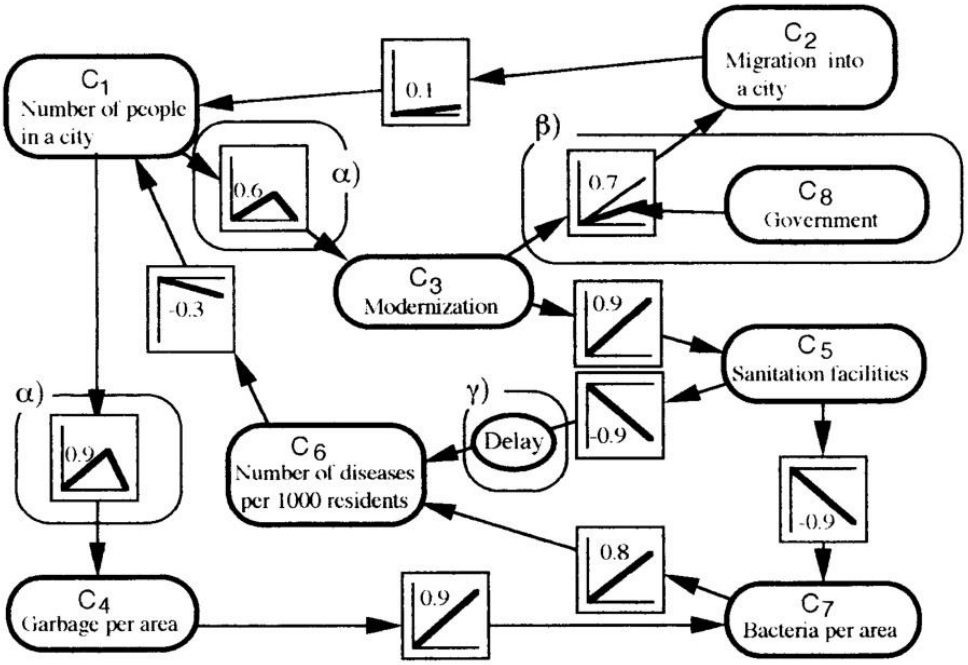


Fig. 26 FCM extended by nonlinear (α), conditional (β) and time-delayed (γ) weights [54].

According to [36], several other extensions and methodologies emerged during the last decade, all of which improve the conventional FCM in various different ways. These include the *Rule Based Fuzzy Cognitive Maps*, the *Fuzzy Grey Cognitive Maps*, the *Intuitionistic Fuzzy Cognitive Maps*, the *Dynamical Cognitive Networks*, etc. Although these methods solve most of the problems and disadvantages outlined in the previous section, they often complicate the process of the FCM design for experts because they increase the number of parameters which need to be adjusted. Furthermore, a mathematical background required to implement solutions using some of these methods is much more demanding, than a simple matrix algebra necessary to employ the conventional FCM.

4.2.3. Three-Term Relation Neuro-Fuzzy Cognitive Maps

In [55] we proposed the *Three-Term Relation Neuro-Fuzzy Cognitive Map* (TTR NFCM) as a novel hybrid method designed particularly to model complex systems. The proposed method enhances the conventional FCMs by adding two new features. The first one is the inclusion of a historical values or trends in the concept update formula using the *Three-Term Relations* (TTR), which are inspired by methods used within the control engineering, namely the PID controllers. As an additional alternative to the TTRs, *time-window* based modification is also proposed. This modification is inspired by similar methods used for prediction and basically changes the concept update formula by including concept activation values from the several previous states instead of

only the last one. The second main feature is the replacement of simple linear weights between concepts by small feed-forward artificial neural networks, which modifies the FCM into the hybrid *Neuro-Fuzzy Cognitive Map* (NFCM).

Our proposal is motivated mainly by the *Extended Fuzzy Cognitive Maps* (E-FCM, see Fig. 26) by Hagiwara [54], but our approach is different and more general. The E-FCMs tackle the problems of the conventional FCMs by adding support for nonlinear relations, conditional relations and time delayed relations. However, the inclusion of these new relation types imposes additional requirements to knowledge of human experts who are tasked to design the E-FCMs. It can be difficult to determine the suitable type of the relation between the specific pair of concepts if the knowledge of the system is limited. In case of the time-delayed relations it is problematic to choose the correct value used for the delay. An application of conditional relations requires an exact knowledge of rules which determine the relations and also knowledge of the mutual dependence of the relations (the conditionals).

The NFCM solves the problems of nonlinear and conditional relations single-handedly by employing FF-ANNs, which are well known to be able to approximate a wide variety of nonlinear functions, including the logical functions such as OR, AND, XOR, etc. The time-delay problems and lack of dynamics within conventional FCM relations is solved by employing TTR or time-window based approaches. Both components of the TTR NFCM can be used together or separately.

4.2.3.1. Nonlinear Relations Represented by FF-ANN

While being easy to understand, visualize and compute, the simple linear weights, which are used within the conventional FCMs, are naturally insufficient for modeling nonlinear relations within complex systems. One deficiency is the linear nature of the relation itself, but the other more important problem is the mutual independence of relations, where each relation acts as a linear function of one variable. The results of these functions are then combined using the concept update formula (10), which is basically the simple implementation of the OR rule. This is the only single rule allowed to determine the mutual dependence between relations in the conventional FCM. By replacing both the linear weights and the concept update process itself by the FF-ANN, we can approximate all possible mutually interconnected relations which are influencing the single concept C_i, as a multivariable function, where input variables depend on the state of all concepts C_j preceding the concept C_i.

The applicability of the FF-ANN instead of conventional weights comes from the fact, that every single cognitive map, even one with several recursive connections, can be unwrapped into multiple

simple feed-forward cognitive maps. These simple FCMs are mathematically equivalent to the former FCM and can be used to compute concept updates. The unwrapping process of the FCM is shown on the Fig. 27.

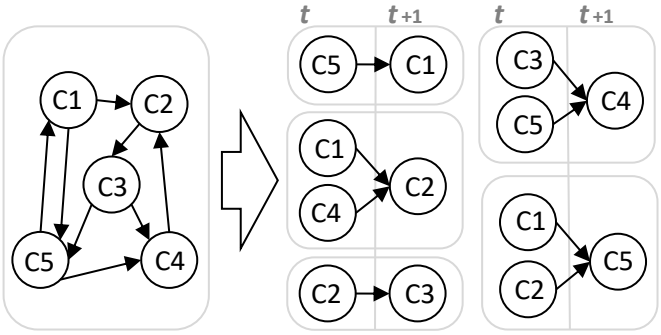


Fig. 27 Example of unwrapping of a simple FCM [55].

With an unwrapped FCM, the possibility for an application of FF-ANNs is clearly evident (see Fig. 28). A rule of a thumb is to replace all relations which are preceding a single concept with an FF-ANN. A number of input neurons in each FF-ANN will be the same as a number of preceding concepts and there will be single output, as there is only one single concept which is being influenced. A topology of the deployed FF-ANNs is to be considered, but generally even small and simple 2-hidden-layer topologies, with less than 5 neurons in each hidden layer, should perform better than the conventional linear FCM relations as their approximation capability is greater [56]. It is also possible to use FF-ANNs with no hidden layer. This way the NFCM will operate equally to the conventional FCM.

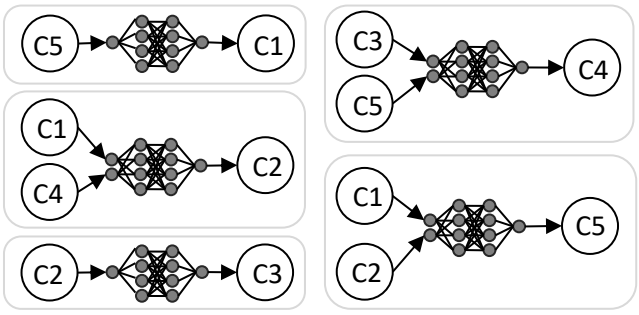


Fig. 28 A proposal for nonlinear relations between concepts using the FF-ANN [55].

The replacement of the linear relations with the FF-ANNs naturally leads to an increased computational complexity of the resulting FCM. This is obvious, as the increased approximation capability cannot be achieved without an increase in the model complexity [56]. The increase in the complexity is equal to the sum of the computational complexities of all used FF-ANNs and grows exponentially with the size of the used FF-ANNs. For small networks this is not the problem. Larger

networks will slow computations down, so it is necessary to find the appropriate size/approximation ratio. However, Fig. 28 clearly shows possible parallelization of computations, which may help to overcome the additional computational overhead caused by computationally demanding FF-ANNs.

The general usage of the NFCM method for system modeling should be the following [55]:

1) The expert designs the FCM by determining the key system variables and defining the corresponding concepts.
2) The expert determines which concepts should be connected (related). No further identification of the type of the relation is necessary as it will be a task for automated learning methods in the next stages.
3) The expert identifies domains of the system variables and declares membership functions for all the concepts. These functions will be used for the fuzzification (or the normalization) of the training data during the learning of the NFMC and for the defuzzification of the concept activations during the runtime.
4) The FCM is unwrapped into smaller feed-forward subsections, all of which are implemented as MLPs and trained using an arbitrary neural network supervised learning algorithm, using the data gathered during an operation of the system.

4.2.3.2. Time Window Relations

To tackle the problem of the time delay and the dynamics of the relations, we propose a *time-window* method (see Fig. 29), which is commonly used for prediction problems. Instead of a single relation between the two concepts, there are T relations, where T determines a size of the used time-window, i.e. the number of previous activation values of the preceding concepts which take part in the concept update. Each k-th activation value $A_j(t–k)$ of the concept C_i preceding the concept C_j has its own weight $w_{ij}(k)$ (or corresponding input neuron in case of using the NFCM). The concept update formula (10) is updated accordingly:

$$A_j(t+1) = p\left(\sum_{k=0}^{T}\sum_{i=1}^{n} w_{ij}(k)A_i(t-k)\right) \tag{14}$$

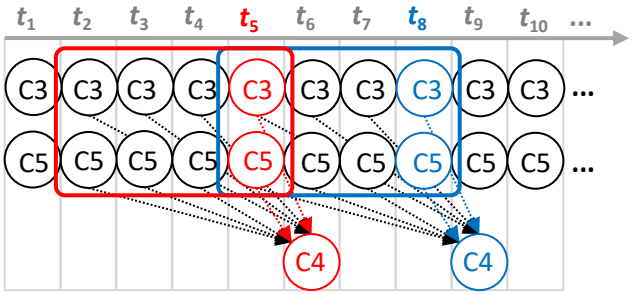


Fig. 29 Visualization of an expansion of the concept update formula by inclusion of states of the concepts from the past (time window) [55].

4.2.3.3. Three-Term Relations

Another proposed solution to the problems of relation dynamics is inspired by methods from the control engineering, namely the PID controllers [57]. We titled the method the *Three-Term Relations* (TTR), since it uses three main components to represent relations between concepts, similarly to the PID controller, which uses three components to modify input variables of the controlled system.

A *proportional-integral-derivative controller*, also known as a PID controller or a *three-term* controller, and its modifications are widely used in a feedback process control within countless industrial control systems [57]. The controller is used to keep a selected system variable within specified limits using the difference between its actual value and its desired value. The controller attempts to minimize the difference (error) $e(t)$ between these values by changing the selected input process variable. The algorithm used to compute the input variable involves three separate components (see Fig. 30) [57]:

- proportional component (P), which depends merely on the present error $e(t)$,
- integral component (I), which accumulates all the error from the past,
- derivative component (D), which is based on the current rate of change of the error.

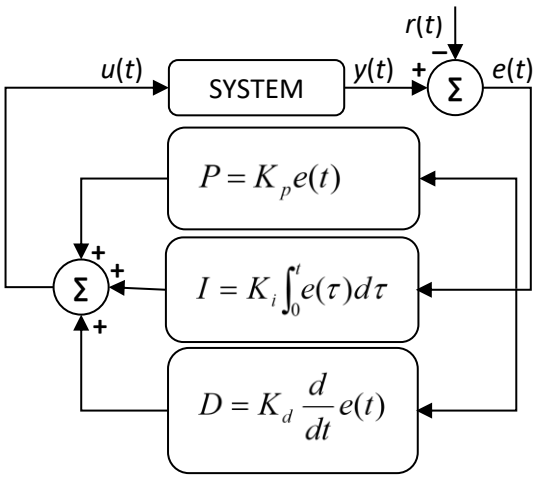


Fig. 30 A block diagram of a PID controller and a controlled system in a feedback loop: $y(t)$ is a value of the controlled system process variable; $r(t)$ is the value of the desired process variable; $e(t)$ is the difference (error) between $y(t)$ and $r(t)$; $u(t)$ is the value of the input process variable computed by the controller. The controller consists of three components – proportional (P), integral (I) and derivative (D).

An advantage of PID controllers is their applicability to a wide range of existing dynamic technical systems due to their ability to quickly react to sudden changes thanks to the derivative component and also to the undesirable long-term trends thanks to the integral component. This is also the main motivation to use a similar approach for FCM modeling. In addition, from the point of view of the control theory, every preceding concept can be seen as a simple controller of the following concept. Actually, a whole PID feedback loop (see Fig. 30) can be represented and also (with slight modifications) implemented as the simple FCM.

For the purpose of utilization within the FCM, we made slight changes to the three components. First of all, we removed the constant tunable parameters K_p, K_i and K_d from the component computation formulae (see Fig. 30) since they are already represented as weights within the FCM. Secondly, we transformed the formulae into a numeric form. Finally, we replaced the integral term with a moving average term, with a goal of diminishing the global stability issues. The reason for this slight modification is the fact, that the computations within FCM are supposed to produce values in range [0,1]. These limits cannot be met using the simple integral term, since it could diverge to positive or negative infinity very easily. Similarly, we also introduced a threshold function p to the derivative term to achieve the same limiting effect. The visualization of the single TTR along with the final component computation formulae is shown on Fig. 31.

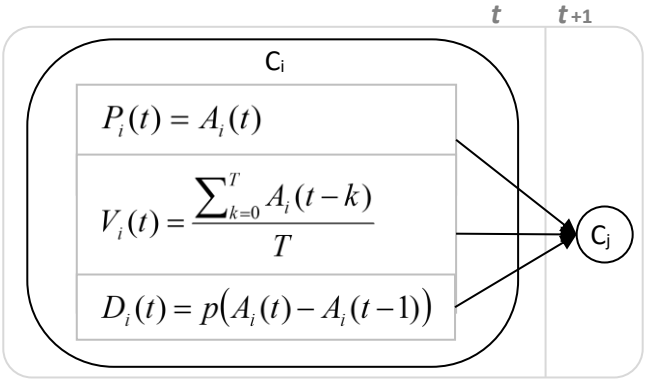


Fig. 31 An example of the three-term relation between concepts C_i and C_j with proportional component P_i, moving average component V_i and derivative component D_i. T is the size of the time window used to calculate the moving average.

With these modifications to the basic structure of the FMC, a computation of a concept updates consists of two stages. Firstly, the component activation values are computed using the formulae from Fig. 31. Secondly, the concept activation values are updated using the expansion of the concept update formula (10), which includes the outputs of all three components within the expression:

$$A_j(t+1) = p\left(\sum_{i=1}^{n} w_{ij}^{P} P_i(t) + w_{ij}^{V} V_i(t) + w_{ij}^{D} D_i(t)\right) \tag{15}$$

While utilizing the TTR, the concepts will react to the last state (or the last activation value of the concept) similarly to the conventional FCM, thanks to the proportional relation component P_i. The additional moving average relation component V_i enables the concepts to react to the long-term trends. This can mitigate short-lived disturbances, noise, or implicit oscillations within the map. The concepts can also react to the sudden changes more quickly than within the conventional FCMs thanks to the derivative relation component D_i. In a similar manner to the time-window approach, there is only single tunable parameter T, which determines the number of previous states considered within the moving average V_i.

The TTR can be also viewed as a simplification of the *time-window* method. While it promises the same responsiveness to the dynamic behavior, it also reduces the number of required relations to only three per concept, which is often far lower than the number used for the time window method.

4.3. Automated Learning of FCM

The disadvantage of FCMs is the fact that, like other types of fuzzy inference systems, they are unable to learn independently and usually use the knowledge of the expert in order to build a model (including set of concepts together with corresponding weight matrix). The design of learning methods is challenging due to the complex structure and variability of FCM. For this reason, a set of concepts *C* is usually given *a priori* by an expert and only the weight matrix *W* is set by automatic learning algorithms [40][43]. The aim of FCM learning is then to find a weight matrix that is best suited to addressing a decision, prediction or other type of problem.

Several automated methods are available for learning of FCM, most of which are based on ideas from the field of artificial neural networks. Machine-learning algorithms can train FCM, i.e. purposefully modify the strength of interconnections (weights) between concepts, similarly to synapses between neurons in neural networks [43]. The suitability of a particular machine learning algorithm depends largely on the type of problem addressed by FCM.

Problems of the *first type* (see chapter 4.1) are addressed through learning approaches based on unsupervised learning, such as *Hebbian learning* [40]. Generally, unsupervised learning is useful, for example, for the role of clustering when we need to sort samples of input data into several groups. When objects within a group have strong relationships, they usually respond or are activated at the same time, which corresponds to the paradigm of Hebbian learning. From the point of view of situational control and modeling, this approach is suitable for finding an appropriate division of the situational space into basic situational frames (classes). It is especially useful when defining or modifying the situational classifier.

In problems of the *second type*, it is better to use supervised learning. These may be, for example, problems of experimental identification, modeling and control of open systems [26], where vectors of input, output, and also internal parameters of these systems are available from experimental data. *Evolutionary* and *population-based optimization* algorithms or least-mean-square-based approaches such *error backpropagation* [40] can be used.

In general, various learning approaches applicable to the adaptation of FCM parameters can be categorized as follows [40][43][46]:

1) Unsupervised learning approaches,
    - based on Hebbian learning (HL),
        - Differential HL, Active HL, Nonlinear HL,

2) Supervised learning approaches,
    - using evolutionary and population algorithms,
        - Genetic algorithm, Simulated annealing, Tabu search,
        - Particle Swarm Optimization,
    - based on the least-mean-square (LMS) method,
        - Delta Rule, Backpropagation of Error, Backpropagation through Time [41],
        - Data Driven Hebbian Learning (DDHL [49]).

For most approaches based on Hebbian learning, a subset of weights matrix is first determined on the basis of expert knowledge, and the goal of further learning is to determine such weights from that subset that lead to FCM convergence into the target region specified by the problem being solved.

In the case of population-based algorithms and LMS approaches (including DDHL), the goal is to automatically calculate the weight matrix using historical data so that the states of FCM match the sequences of input state vectors or patterns. In this case, the expert is replaced by historical data [3][9].

In the next sub-chapters, we continue and expand the ***first contribution*** of this dissertation (*"the modification of the basic concept of FCM in order to allow the modeling and control of complex systems"*) with addition of novel methods applicable to *adaptation* of FCM parameters.

4.3.1. Interactive Evolution of Fuzzy Cognitive Maps

An interesting approach to adaptation of system parameters is to use *interactive evolutionary optimization* (IEO) [58]. This approach revolves around specific definition of fitness function of an individual during evolution. The fitness function usually incorporates two basic types of evaluation criteria: *qualitative* and *quantitative*. Conventional evolutionary systems utilize *quantitative criteria*, usually represented by analytic (fitness) function. However, there are problems which cannot be evaluated using analytic functions or their formulation is strongly subjective. Individual solutions are hence evaluated using *qualitative criteria*, such as taste, beauty of images, quality of sounds etc [59]. Proper evaluation of such criteria requires the incorporation of a *human evaluator* (or expert) into the optimization process (see Fig. 32) which is the core idea of IEO.

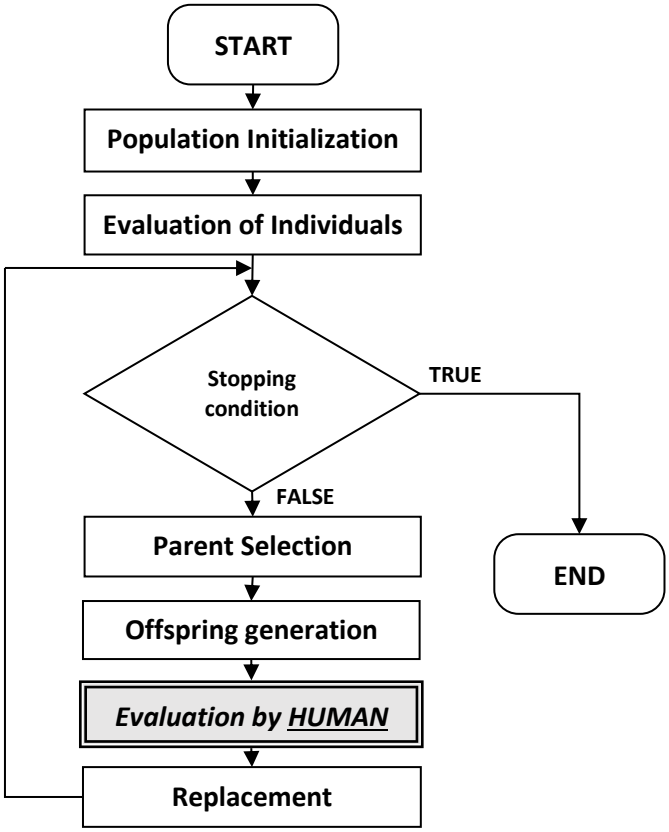


Fig. 32 Interactive evolutionary optimization.

A human evaluator subjectively determines the quality of individual solutions using his internal psychological fitness landscape (see Fig. 8 in chapter 2.1.2) and IEO drives the solutions towards the global optimum according to the psychological distance [61]. In order to perform the evaluation, an evaluator is often provided by a graphical user interface (GUI) which offers possible solutions in a proper form [59]. There are several options for GUI implementation (such as *number* of individuals displayed, display of *parents vs offsprings* or *offspring only*, method to *evaluate* an individual, etc.). For example, the evaluation itself can be performed using various approaches, such as:

1) assignment of simple (crisp) value to each solution (*direct fitness setting*),
2) assignment of interval or fuzzy value to each solution (*approximate fitness*)
3) selection of best solution from a list of solutions (*direct parent selection*),
4) manual sorting of solutions from the best to the worst (*manual population sorting*),
5) hybrid methods combining analytical fitness function and human evaluation (*semi-interactive evolution*, see chapter 4.3.2),
6) other combined methods [62].

The form and offered functionalities of GUI play a crucial role in the success of the optimization process [59]. It is not only a question of graphical user-friendliness. This problem is further discussed and reviewed in [62].

In [59] we proposed a novel algorithm based on IEO principles titled *Interactive evolutionary optimization of fuzzy cognitive maps* (IEO-FCM). We applied the method for adaptation of FCM controller of a mobile robot driven on a simulated racing track. The adaptation process is based on IEC, evaluation GUI is shown on Fig. 33. More detailed description of FCM design can be found in paper [59]. A simulator software implemented in Python is available at [60].

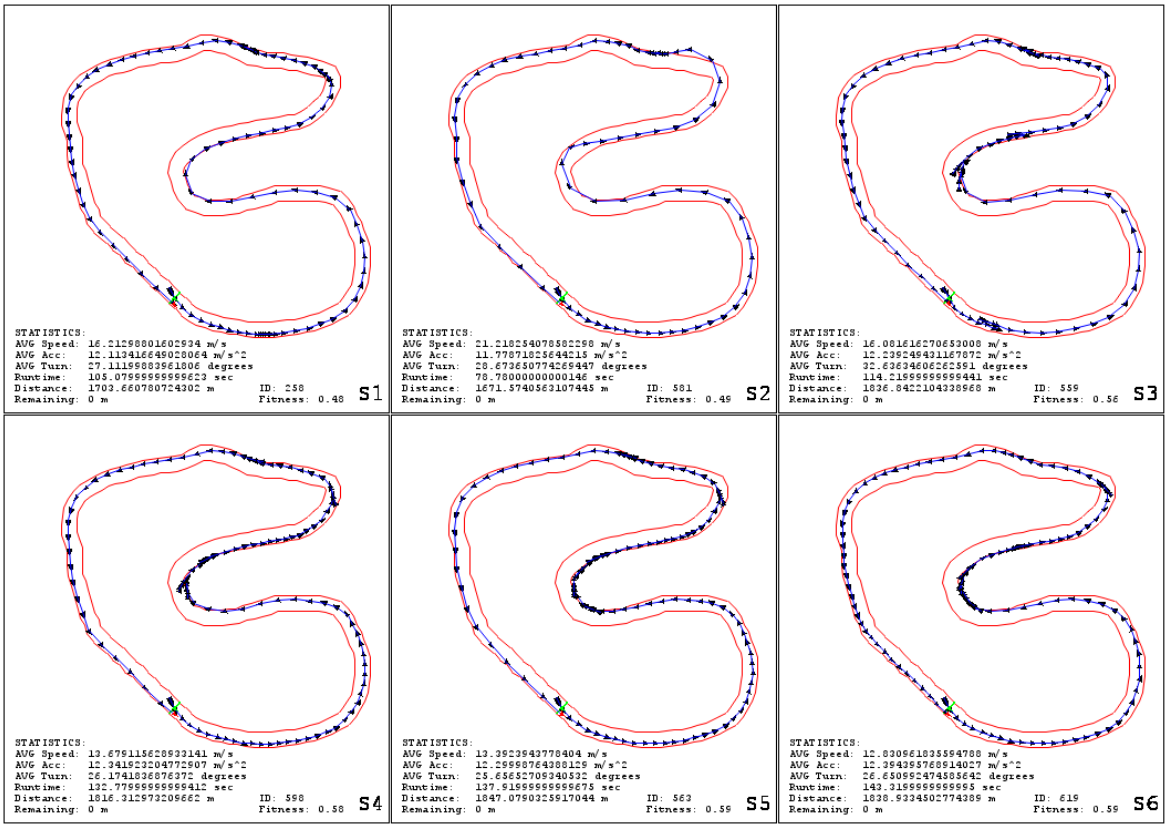


Fig. 33 Example of graphical user interface used for interactive evolution of FCM controller. The interface shows several individuals from a single generation of population which are to be evaluated by an expert. The expert is provided by complementary information and statistics about each solution to help him with the decision. In the semi-interactive mode, the automatically computed fitness function is displayed as well, however, the value may be changed according to the assessment of the expert [59].

The controller adaptation process was based on quantitative as well as qualitative criteria [59]:

- C1: relative average absolute angular turning,
- C2: relative average absolute acceleration,
- C3: relative trajectory length,
- C4: relative average speed,
- C5: relative road passing time,
- C6: number of sheers (deviations from road borders),
- C7: success or fail in passing the road (yes or no).

The criterion C1 represents the average angle of the robot trajectory related to the center of the track. The smaller the value is the more stable the solution is. The criterion C2 represents relative average acceleration or deceleration. Again, lower value is better since control is smoother and more stable. The criterion C3 represents the total length of the trajectory (lower is better). The criterion C4 represents the average speed (higher is better). The criterion C5 represents the time necessary to complete the circuit (lower is better). The criterion C6 shows the number of collisions, which should be prevented (lower is better) [59].

Several controller designs created as a result of our IEO-FCM method were compared to a neuro-fuzzy system AN-FIS, whose parameters were adjusted using a *self-organizing migrating algorithm* (SOMA) as a special kind of EA [63]. The Tab. 2 shows the results of the five most successful FCM designs evolved using IEO and fulfilling the stopping criterion C7. The criteria C1 – C6 are mutually compared for FCM and AN-FIS controller [59].

Tab. 2 Relative comparison of experimental results of FCM and AN-FIS controllers [59].

Controller	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
FCM 1	0,756	0,978	0,982	0,664	0,871	1
FCM 2	1	0,937	1	**1**	**0,601**	**0**
FCM 3	0,825	1	0,984	0,816	0,713	1
FCM 4	0,853	0,961	0,978	0,781	0,753	1
FCM 5	0,783	0,978	**0,974**	0,743	0,777	**0**
AN-FIS	**0,553**	**0,417**	**0,974**	0,585	1	**0**

As we can see, the results of FCM and AN-FIS are quite comparable. From a population of 30 individuals the first five ones reached a very good quality, although AN-FIS controller seems to be slightly better visually, see Fig. 34. For a better imagination, the best results in the table are

boldfaced. We can see that FCM 2 controller is the best in three criteria (C4, C5 and C6) and AN-FIS controller shows the best quality in four criteria (C1, C2, C3 and C6) and so these two controllers seem to be complementary regarding their quality [59].

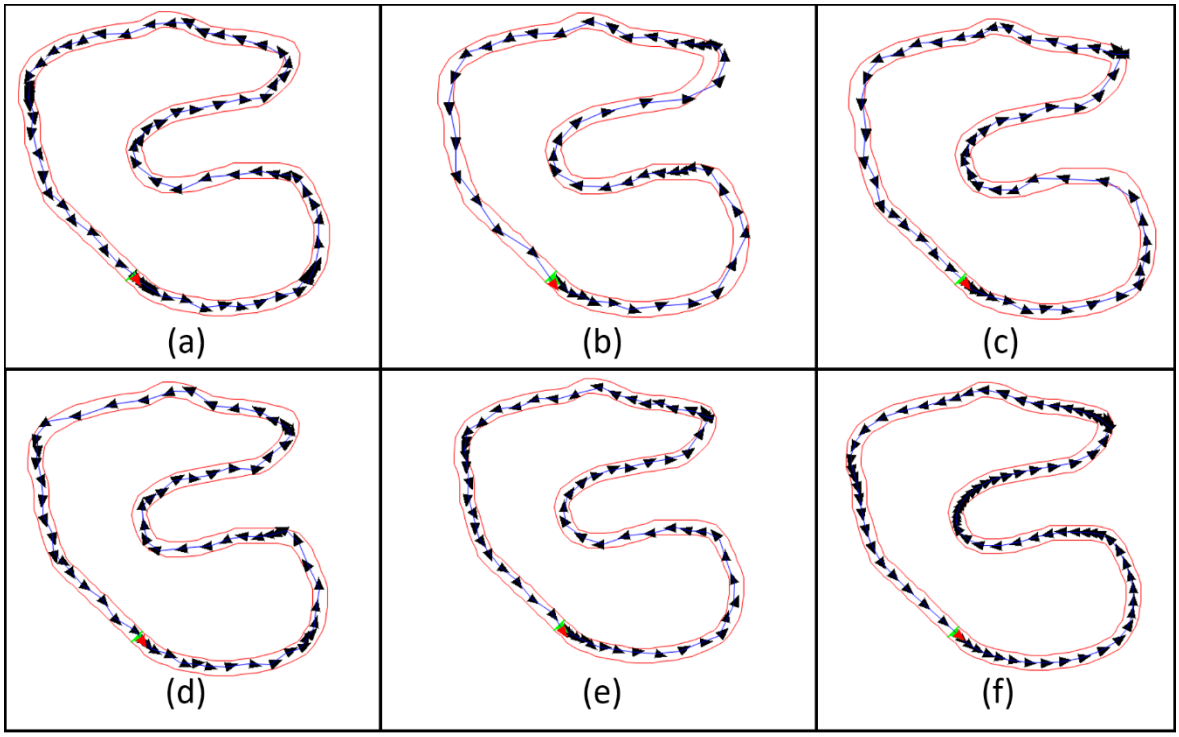


Fig. 34 Final movement trajectories for the best controller designs: (a) - FCM 1, (b) - FCM 2, (c) - FCM 3, (d) - FCM 4, (e) - FCM 5, (f) - AN-FIS [59].

4.3.2. Semi-Interactive Evolution of Fuzzy Cognitive Maps

The main advantage of IEO is its application even in case when it is impossible to express the fitness function in an exact analytical form. However, there are also some drawbacks, where the most obvious one is *human fatigue*. The evaluation process during IEO is very time demanding for a human evaluator, who may be subject to exhaustion and consequent errors, which is obviously undesirable.

Normally, the strength of traditional *non-interactive EA* is tied to enormous populations being evolved through thousands, or even tens of thousands of generations. In any case, it is difficult to evaluate such a massive population with human evaluators. An ordinary population size of *interactive EA* is severely restricted in the quantity of individual solutions showed on a GUI interface, therefore the quantity of individuals in a single IEC generation usually does not surpass 10–20. Obviously, such a restriction does not ensure an acceptable exploration of the search space.

However, there are cases of optimization problems, where it is possible to specify at least some (usually weak) heuristic function, and even though it may not be efficiently used in order to find the

solution non-interactively, its combination with human evaluator may significantly speed up the optimization process.

With this assumption, we proposed another novel approach [64] which combines *automatic heuristic optimization* with *interactive optimization* into hybrid *semi-interactive evolutionary optimization* (S-IEO). The main goal is to speed up the optimization process and at the same time also reduce the expert involvement and fatigue. The S-IEO changes the typical IEO algorithm (see Fig. 32) in these areas:

- The human evaluator is provided with a *complementary heuristic function*.
- The human evaluator may let the evolution run automatically, *without intervention*, provided that the current progress is considered satisfactory.
- However, he may *intervene as soon as it seems appropriate*.
- Or else, he may decide to *intervene regularly*, i.e. every 10th generation in order to remove individuals which he considers unfit for further evolution.

The S-IEO assumes the following contributions of the method:

- Ability to use potentially *weak and inaccurate heuristic* as a fitness function.
- Ability to use significantly *larger populations* than ordinary IEO.
- *Reduction of evaluator's fatigue*, since he does not need to evaluate every single individual by hand.
- Decrease of *time spent on evaluation* of individuals.
- Decrease of *overall optimization runtime* (and general *speed-up* of evolution).

As an example of a complementary fitness function, we may consider the criteria defining the success of a mobile robot on a racing track from the previous chapter (displayed also in Tab. 2). In this case, we may propose the following complementary fitness function:

$$F_i(C1, C3, C4) = \frac{|l - C3_i|}{l} + \frac{1}{C4_i} + \frac{C1_i}{\pi/2} \tag{16}$$

where l is the length of the track, i denotes single individual solution, $C3_i$ is its trajectory length, $C4_i$ is average speed and $C1_i$ is average turn angle. The lower the value of $F_i()$, the better the solution is.

It is clear, that the given function does not cover all the criteria (and other runtime statistics) and obviously may lead to individual solutions which differ significantly from the desired goal. For example, the individual may go straight with maximum velocity and then suddenly stop when

the length of its trajectory equals the track length l. This is fair in order to optimize $F_i()$ but far away from a solution which completes the circuit properly. Such and similar shortcomings can be mitigated by employing additional automatic control mechanisms implemented in the simulation, which will stop the execution of such individuals.

We have performed experiments using this complementary fitness function in three modes:

a) *Non-interactive mode* – where human evaluation was disabled and proper circuit completion was achieved by enforcing additional control mechanisms (collision control, wrong way warning, deviation detection, etc.).
b) *Fully-interactive mode* – in which the human evaluator assessed all the solutions by hand in every single generation.
c) *Semi-interactive mode* – in which the human evaluator assessed the solutions only every 10^{th} generation.

The experiments focus on the assessment of contributions brought in by S-IEO, namely the speedup of optimization, decrease of time spent on evaluation of solutions and overall decrease of evaluator fatigue. Four time intervals were measured within each experiment:

1) *Total program runtime* – time elapsed from the start of the program to finish.
2) *Evolution runtime* – time spent on simulation of individual trajectories.
3) *Waiting time* – time during which the program is idle, waiting for evaluator's interaction.
4) *Interaction time* – time during which the evaluator is interacting with program GUI.

In order to obtain more comparable results, the experiments in each mode were performed 3 times (each from the start of the evolution until the fulfillment of the finishing criterion) and then averaged. The results of the *non-interactive evolution* are displayed in Tab. 3, the results of the *fully-interactive evolution* are shown in Tab. 4 and the results of the proposed *semi-interactive evolution* are displayed in Tab. 5. Summary of relative comparative results of individual modes is shown in Tab. 6. Good results are highlighted in green, average results are yellow and worse results are shown in red.

Tab. 3 Results of the non-interactive evolution of FCM controller [64].

Non-Interactive	1st run	2nd run	3rd run	Averages
runtime:	52 m 55,206 s	36 m 6,968 s	24 m 15,095 s	37 m 45,756 s
evolution:	52 m 54,080 s	36 m 6,261 s	24 m 14,387 s	37 m 44,909 s
waiting:	0 m 0,000 s	0 m 0,000 s	0 m 0,000 s	0 m 0,000 s
interaction:	0 m 0,000 s	0 m 0,000 s	0 m 0,000 s	0 m 0,000 s
generations:	45	35	22	34
individuals:	930	730	470	710
best:	0,3373773	0,2954482	0,3863609	0,3397288
interactivity:	Disabled	Disabled	Disabled	Disabled

Tab. 4 Results of the fully-interactive evolution of FCM controller [64].

Full-Interactive	1st run	2nd run	3rd run	Averages
runtime:	50 m 20,218 s	22 m 25,855 s	24 m 51,087 s	32 m 32,387 s
evolution:	19 m 10,607 s	6 m 55,316 s	17 m 37,290 s	14 m 34,404 s
waiting:	7 m 9,296 s	11 m 10,903 s	2 m 10,033 s	6 m 50,077 s
interaction:	23 m 59,590 s	4 m 18,950 s	5 m 3,066 s	11 m 7,202 s
generations:	42	12	18	24
individuals:	870	270	390	510
best:	0,6926282	0,3709367	0,4101237	0,491229533
interactivity:	Enabled-1	Enabled-1	Enabled-1	Enabled-1

Tab. 5 Results of the semi-interactive evolution of FCM controller [64].

Semi-Interactive	1st run	2nd run	3rd run	Averages
runtime:	21 m 58,540 s	25 m 52,580 s	17 m 34,025 s	21 m 48,382 s
evolution:	14 m 21,396 s	18 m 54,354 s	15 m 17,751 s	16 m 11,167 s
waiting:	1 m 37,144 s	3 m 43,171 s	1 m 0,872 s	2 m 7,062 s
interaction:	5 m 59,259 s	3 m 14,350 s	1 m 14,689 s	3 m 29,433 s
generations:	45	25	17	29
individuals:	930	530	370	610
best:	0,8836055	0,4626138	0,4640605	0,6034266
interactivity:	Enabled-10	Enabled-10	Enabled-10	Enabled-10

Tab. 6 Relative comparison of results of all IEO methods [64].

Adaptation Method:	Non-Interactive	Semi-Interactive	Full-Interactive
Program runtime:	1	0,577459124	0,86169313
Evolution time:	1	0,428788467	0,386065932
Waiting for user:	0	0,309849687	1
Interaction time:	0	0,313896941	1
Num. of generations:	1	0,852941176	0,705882353
Generated individuals:	1	0,85915493	0,718309859
Best fitness:	0,381818923	1	0,783865877
Interaction interval:	Disabled	Every 10 generations	Every 1 generation

The comparative results fully comply with the assumed contributions of S-IEO (on page 71):

- The overall *program runtime* of S-IEO was decreased by 42 % compared to the non-interactive evolution and by 14 % compared to the fully-interactive evolution.
- The *evolution time* of both S-IEO and the fully-interactive version is comparable but significantly lower (by approx. 60 %) to that of the non-interactive version, since without human evaluator it has to explore significantly higher portion of the search space.
- Very promising are the results tied to *human fatigue*. It is clear, that both the unused *waiting time* and also the *interaction time* are lower by almost 70 % in favor of S-IEO.
- The *total number of generations* and total *number of generated individuals* is slightly lower in the fully-interactive version, however it is questionable if this is the improvement, since S-IEO evaluated the higher number of individuals in only one third of the time compared to the fully-interactive evolution and hence covered a larger portion of the search space.
- The *fitness value* of the final solution is naturally best in the non-interactive version, since it is primarily focused on individuals satisfying the fitness function while *ignoring the opinion of human* evaluator. Considering that the fitness function is used only as a complementary heuristic, it should *not be used* for *final evaluation* of the objective quality of resulting solutions. It should be noted, that *all final solutions* of all three methods *achieved the finishing criterion*, that being the ability to finish the course without collision.

5. Intelligent Space as Complex System

"Every generation imagines itself to be more intelligent than the one that went before it, and wiser than the one that comes after it."

George Orwell

The enduring proliferation of *networking technologies*, *ubiquitous sensors* and *artificial intelligence* into all areas of human activity has given a rise to many novel ideas including the *internet of things* and its specific manifestations, often marketed as *intelligent space*.

The global *Internet* has changed our world in ways that would be previously unthinkable. Originally, its development began slowly. However today, the innovation and progress are happening at an exceptional rate. From humble beginnings as the *Advanced Research Projects Agency Network* (ARPA-NET) in 1969, connecting a handful of computers, it is now expected to interconnect 75 billion devices by 2025 [65]. The overall *internet of things* market is projected to be worth more than one billion U.S. dollars annually from 2017 onwards [65].

In this chapter, we present the most up-to-date development in *networking*, *cloud computing*, *fog computing* and related technologies, specifically aimed towards practical applications of the *internet of things* and *intelligent spaces*. We further argue, that intelligent spaces as complex systems can be managed and controlled using methodologies, algorithms and tools proposed in previous chapters, including the *situational control* by *fuzzy cognitive maps*.

5.1. Internet of Things

The *Internet of Things* (IoT) is a recent paradigm that aims to cross barrier between the physical and digital world. The main idea is to "integrate the state of the things that form the world into software applications, making them benefit from the world's context information" [68]. IoT allows common everyday objects in our environment to become active participants in mutual communication with people, by sharing information with other members of IoT network, and to be able to recognize events, changes in their surroundings and respond to them in an appropriate manner autonomously.

IoT "considers pervasive presence in the environment of a mutually interconnected devices that are able to interact and cooperate with each other in order to create new applications or services and reach common goals" [66]. The interaction and cooperation are based on certain types of communication which can be either [66]:

- *Human to Machine* (H2M),
- *Machine to Human* (M2H),
- *Machine to Machine* (M2M).

With regards to its development, the IoT should not be considered as some miraculous radical change, or a revolutionary new paradigm, but rather as a gradual evolution of technologies, networks and associated services (see Fig. 35). While initially, the internet established itself as a tool providing humans with a means of communication (via email services) and access to information (via www services), it progressively expanded by adding new services and applications related to the real world (web applications, e-shops, information systems, etc.). This expansion of available internet services is often referred to as *Web 2.0* and culminated with a rise of social media and peer to peer technologies, which empowered ordinary users with a capacity to not only *consume*, but also to *produce and share content*. The IoT proceeds with this "internet-to-real-world" interconnection paradigm by providing the same *content production* and *sharing* capabilities to *ordinary devices*.

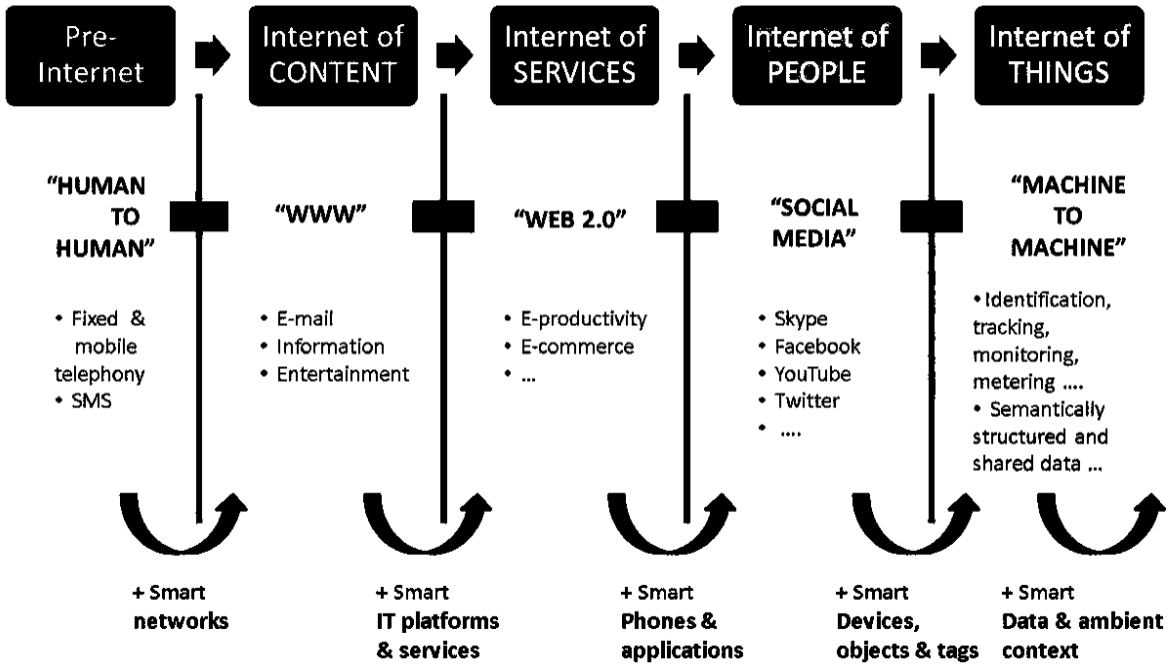


Fig. 35 Evolution of the Internet of Things [69].

5.1.1. Communication in IoT

It is obvious, that computer networks may connect devices with various hardware architectures and operating systems. However, even in spite of variability of hardware and software, the communication and data exchange between different devices can be established using recognized

communication protocols. The network protocol is a norm comprising of rules, formats and procedures designated for a data exchange (including connection initiation, communication and data transfer). Since the matters of inter-computer communication are so complex, that they cannot be not described by a single protocol, a whole system of protocols has been established by the *International Organization for Standardization* (ISO). The proposed reference model was marked as *Open Systems Interconnection* (ISO/OSI) [78] and serves as a basic framework for communication in contemporary computer networks. It has separated network communication in several layers based on the following principles:

- Each layer should provide precisely defined functions, which need to be selected in such a manner, that its implementation can be realized using standardized protocols.
- The interfaces between the layers need to be selected in order to minimize the data flow between the layers.
- The number of layers should be so large, that mutually distinctive functions does not need to be included in the same layer, as well as so small, so that the whole architecture will remain comprehensible enough.

The established ISO/OSI model consists of seven layers (see Tab. 7), which participate on network communication. The model further describes functions of each layer, where the bottom layers provide services to the layers above.

Tab. 7 Open Systems Interconnection (ISO/OSI) model [79].

Layer		Protocol data unit (PDU)	Function
Host layers	7. Application	Data	High level APIs, resource sharing, remote file access.
	6. Presentation		Data translation between application and networking service; data encoding, compression and encryption.
	5. Session		Communication sessions, continuous exchange of information between two nodes.
	4. Transport	Segment Datagram	Reliable transmission of data segments between nodes, including segmentation, confirmation and multiplexing.
Media layers	3. Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control.
	2. Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer.
	1. Physical	Symbol	Transmission and reception of raw bit streams over a physical medium.

The OSI model has established itself as a tool for explaining the operation of networking in general terms. It has been widely used as an educational tool and also to help to describe interactions between the components of other protocols and even hardware networking devices. While many technologies were not designed specifically to meet the expectations of the OSI model, most of them have been further redefined in order to fit into OSI layers. This includes networking protocols, software applications, and even different types of networking devices, such as switches and routers. The model is also useful to those who develop software and hardware products, by helping to make clear the roles performed by each of the components in a networking system [80].

While the ISO/OSI is being a well-recognized and accepted formal model, in practical applications, other models are applied. Currently, the majority of the Internet uses a *TCP/IP model* (see Fig. 36). TCP/IP can use a wide array of various protocols, but the two of them are considered the most important and hence give a common name to the model. The *Internet Protocol* (IP) is roughly equivalent to the third layer (the network layer) of the OSI model and provides addressing, datagram routing and other functions. The *Transmission Control Protocol* is used on the fourth layer (the transport layer) and is responsible for establishment and connection maintenance and reliable data transfer between devices [80].

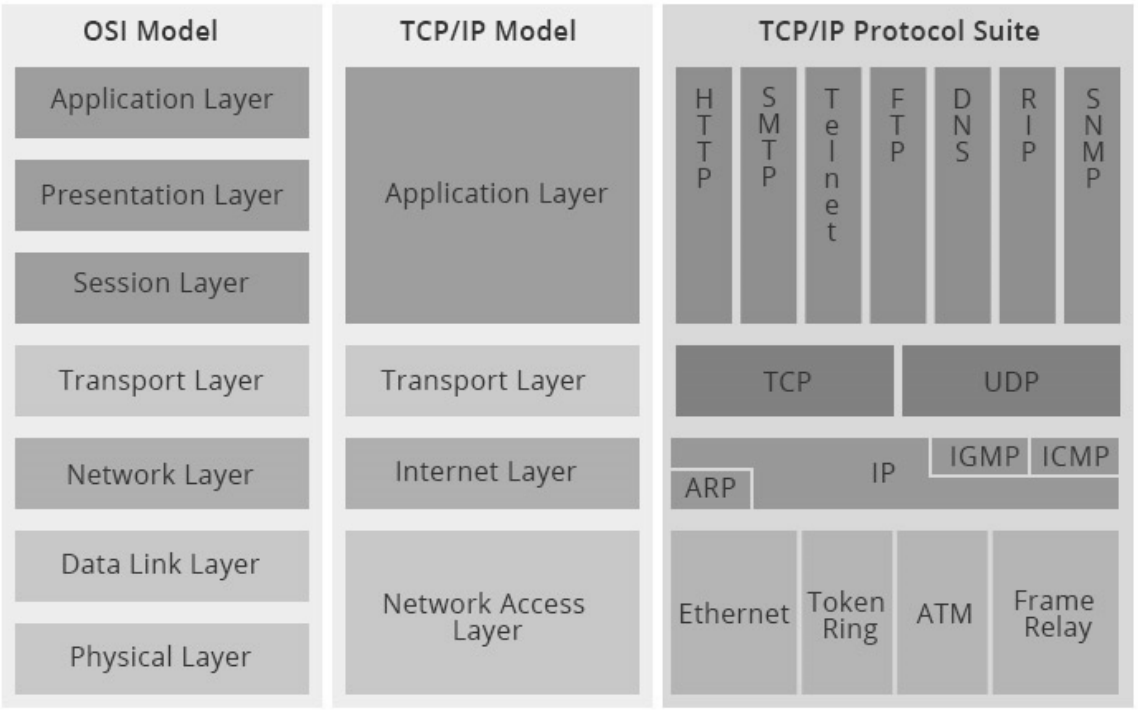


Fig. 36 Comparison of OSI and TCP/IP models including applicable protocols [70].

There are several versions of both TCP and IP applied in active use. The most wide-spread version of IP is its fourth version *IPv4*, which uses 32-bit addressing and is theoretically able to identify about 4.3 billion unique public addresses. With a rising number of connected devices, this limit has currently become a problem for further development of the Internet. In order to bypass the issue, temporary solutions have been proposed, including the use of a subnetting or the *Network Address Translation* (NAT), which deploys one public IP address in order to identify a whole network of devices. However, this approach is not very convenient for IoT applications, because it is not trivial to connect devices inside a subnet, since all of them have a single public address. Therefore, a newer, sixth version *IPv6* is currently being applied. It uses 128-bit addresses with 3.4×10^{38} total possible combinations, that should be sufficient for the foreseeable future.

The most upper layer of the TCP/IP suite (which is roughly equivalent to the fifth-to-seventh layers of the OSI reference model, see Fig. 36) employs a protocol directly tied to a specific *application* or a network service, e.g. email, web, filesharing and other. Without any doubt, the most used internet service is a *World Wide Web* (www), which applies the *Hypertext Transfer Protocol* (HTTP) or its newer, cryptographically *secured* expansion (HTTPS). HTTP enables a connection between clients and servers using a request-response communication. It uses several methods for creating requests, including GET, POST, PUT, HEAD, DELETE, PATCH and OPTIONS. The most utilized methods

are GET, which is used to request data from a specific resource on a server, and POST, which is used to send data to a server or upload a resource.

Since www-related services are so prevalent, the HTTP protocol has started to be used in new, previously unconsidered ways, especially with regards to the rise of IoT. HTTP requests can be used to send commands to various IoT devices, to receive information from sensors and to create interfaces for human-computer interaction. HTTP is frequently used for creation of *Application Programming Interfaces* (API), which can be used with *Cloud* and *Fog Computing* services. More about these will be discussed in the next chapter.

The last part of the TCP/IP suite, which we have not mentioned yet, is the *Network Access Layer*, roughly equivalent to the first and second layer of the OSI model. It deals with the underlaying *hardware* and physical connection used to transfer data. The communication can be established using various technologies, including:

- wired connection (usually via the Ethernet-based family of networks) provided over:
    - coaxial cable,
    - twisted pair copper cable,
    - or optical fiber.
- wireless connection provided over:
    - Wi-Fi (IEEE 802.11)
    - Bluetooth (IEEE 802.15.1)
    - ZigBee (IEEE 802.15.4)
    - Low-Power Networks / LP-WAN (Sigfox, LoRa, Ingenu, Nwave etc.)
    - Cellular networks, etc.

Naturally, this list is not exhaustive and there are many more applicable technologies (e.g. NFC, RFID, satellite and so on). The most important properties of each technology are *transmission rate*, *communication range*, *deployment costs* and *energy consumption*. Typically, the *wired connections* have the *best transmission* rates (over 10 Gb/s) and offer theoretically unlimited range, however suffer from high deployments costs and energy consumption. The properties of wireless technologies vary greatly. *Wi-fi networks* provide some of the highest transmission rates (~1 Gb/s), but offer *low range* (20-30 m indoor) and are quite *energy demanding*. The *Low-Power Networks*, on the other hand, provide *great range* (up to 50 km) and *low energy consumption* but the transmission rate is exceptionally low (only 0.1 Kb/s with Sigfox network). The cellular networks stand somewhere in between Wi-Fi and LP-WAN, but benefit from *wide availability* [90].

5.1.2. Information Processing of IoT

Current progress of IoT is related very tightly to the advancement and application of *cloud computing* tools and services. According to U.S. National Institute of Standards and Technology (NIST), cloud computing is defined as [81]: "*a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services that can be rapidly provisioned and released with minimal management effort or service provider interaction*."

Cloud computing is characterized by on-demand services provided as needed without requiring human intervention, as well as broad network access, resource pooling which serves multiple consumers at once, and rapid elasticity where services scale according to the demand. Cloud services are also often metered, providing transparency of utilized resources to both the consumer and provider of the service [81]. The services are provided in various levels of user autonomy (see Fig. 37).

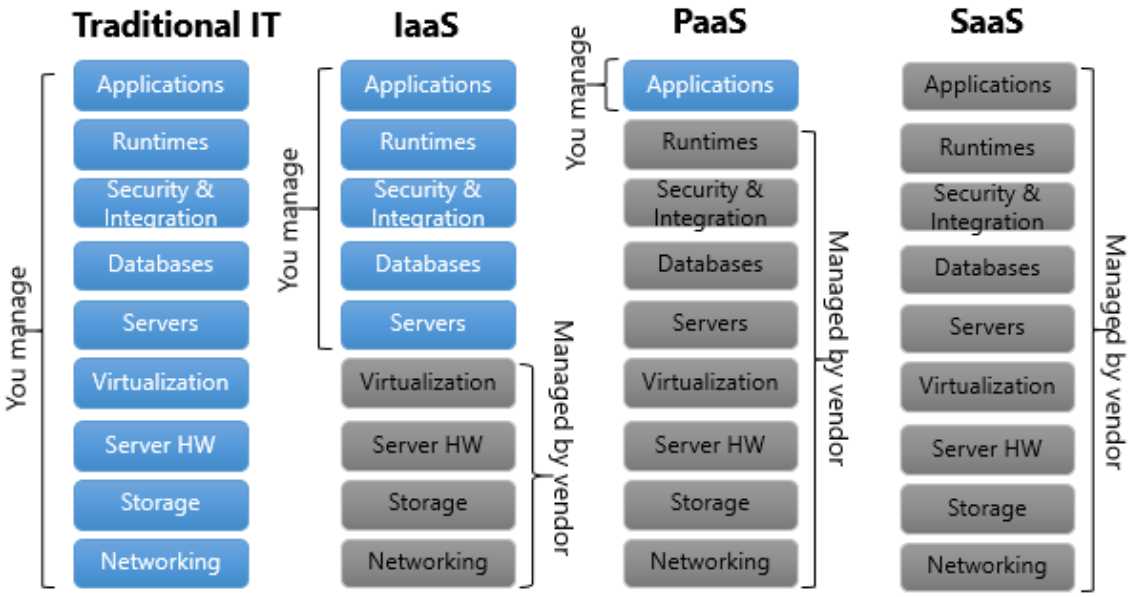


Fig. 37 Cloud computing services compared to the traditional IT [82].

Software as a Service (SaaS) provides the consumer devices with applications running on a cloud infrastructure. The applications are usually available through an established web API, or graphical user interface. The consumer does not manage anything, with an exception to specific application configuration [81]. Typical SaaS applications include AI services, such as speech-to-text, video processing and image recognition, storage management and office tools.

Platform as a Service (PaaS) enables the consumer to deploy any application created using programming languages, libraries and tools supported by the provider. The consumer cannot control the underlying infrastructure, but has the ability to install applications and configure the application-hosting environment [81]. The most common PaaS incorporate environments of popular cloud service providers including MS Azure, IBM Cloud (previously Bluemix) and Amazon Web Services.

Infrastructure as a Service (IaaS) provide the consumer with the most complete control, at least compared to the previous service models. The consumer is able to select and configure all the processing, storage, networking and other fundamental computing resources and deploy and run arbitrary software including full-scale operating systems [81]. These services are often provided by the established ICT companies such as Cisco, IBM, and Intel as an outsourcing solution to a new rapidly growing enterprises, which would otherwise be constrained to the less dynamic, traditional IT infrastructure.

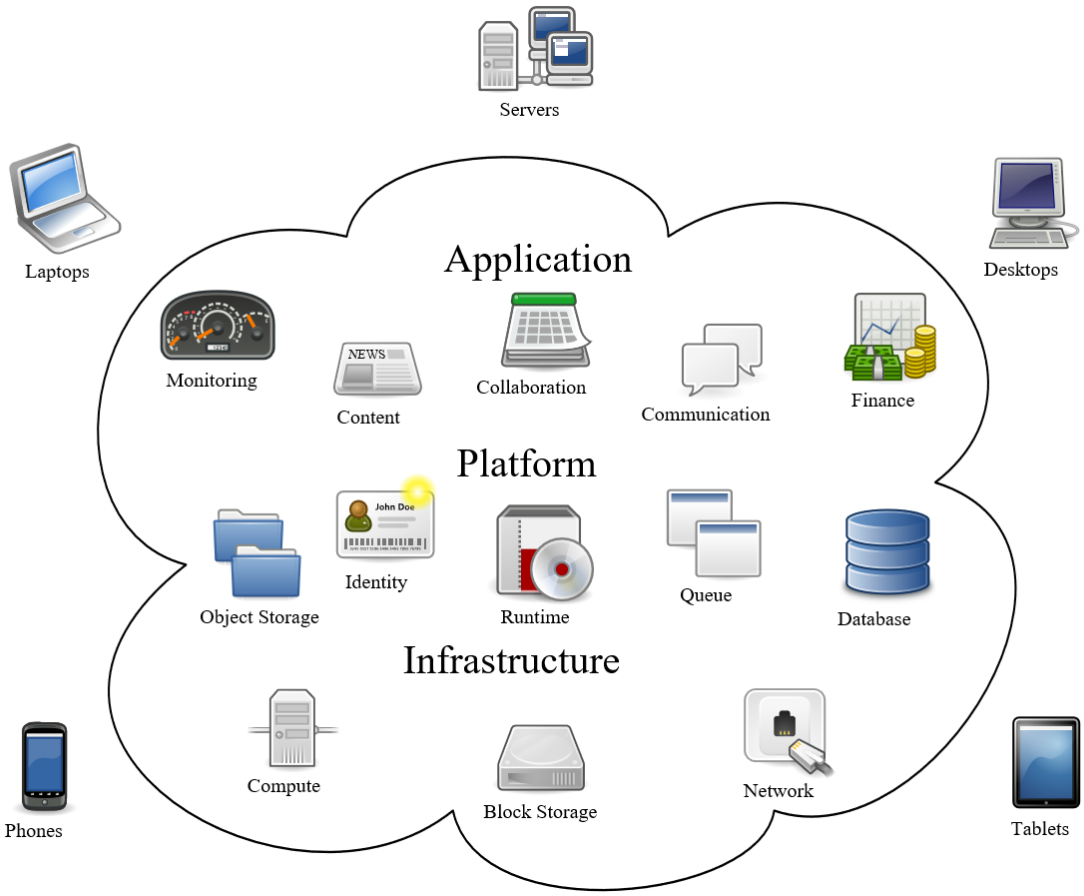


Fig. 38 Typical applications supported by cloud computing model [77].

In terms of IoT, cloud services enable devices to store data or access programs over the internet instead of using their respective (often limited) local storage and computing capabilities. Further, online resources remove the need for the user to be in the same physical location as the hardware that either produce, utilize or store data, since everything is managed and available remotely. However, the cloud solutions may not be suitable in all cases, specifically when transferring and processing large amounts of data. The solution to this issue can be found in application of *fog computing* closer to the edge of the network (see Fig. 39).

The term *fog computing* was established by Cisco as an extension "*of the cloud computing paradigm from the core of network to the edge of the network*" [84]. The definition was further specified by U.S. NIST [86], which defined it as "*a layered model for enabling ubiquitous access to a shared continuum of scalable computing resources. The model facilitates the deployment of distributed, latency-aware applications and services, and consists of fog nodes (physical or virtual), residing between smart end-devices and centralized (cloud) services. The fog nodes are context aware and support a common data management and communication system. They can be organized in clusters – either vertically (to support isolation), horizontally (to support federation), or relative to fog nodes' latency-distance to the smart end-devices*" [86].

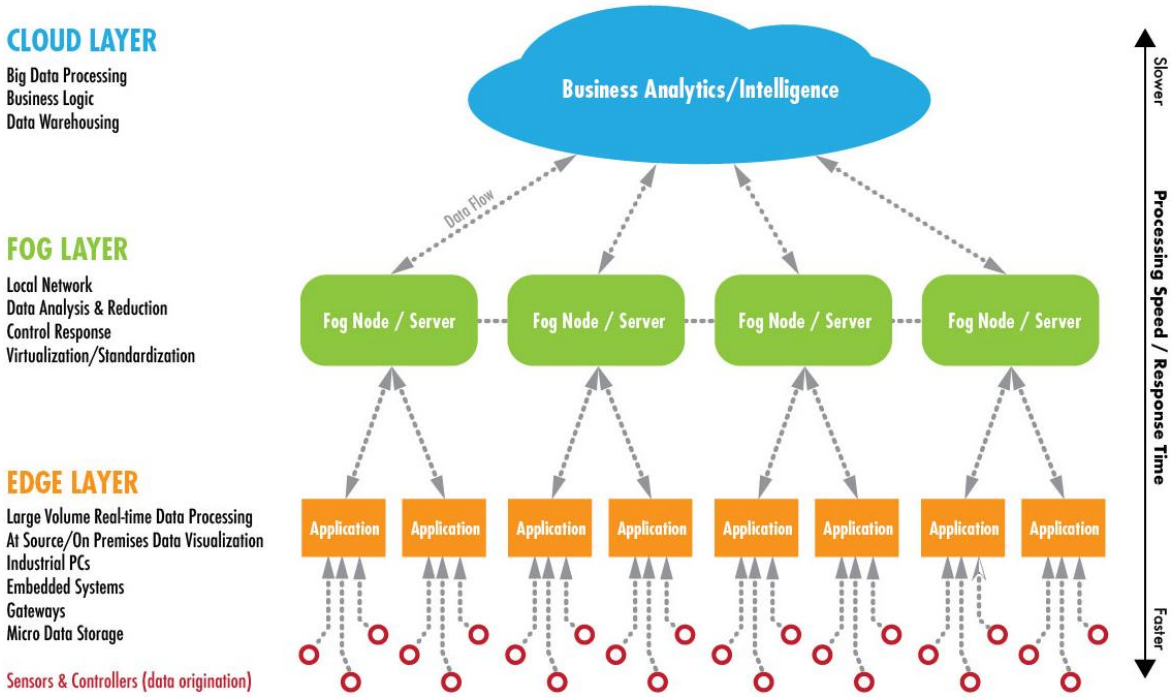


Fig. 39 IoT data processing layers (cloud, fog and edge) [83].

Fog computing provides intermediary computation, storage, and networking services located between end devices and traditional cloud servers [84]. It's a hybrid system-level architecture paradigm where the possibilities of cloud computing, distributed processing and analytics power are brought closer to the edge of a network. It does so in a different way than *edge computing*, which aims to bring the intelligence, analytics, computing, communications etc. directly into the devices at the edge. Fog computing is designed to deal with the challenges of cloud-based IoT solutions by managing IoT data generated by sources along its the route to the cloud servers. It does so by decentralizing data analytics but also applications and management into the network with its distributed and federated compute model [85].

In simple terms, instead of transporting all data over the network and then processing it in the cloud, some operations are performed closer to the IoT devices. Fog computing offers many benefits, such as [85]:

- reducing the cost associated with high bandwidth utilization,
- solving high latency on the network,
- reducing risks of network connection issues and bottlenecks,
- providing higher speed of analysis and action, etc.

5.1.3. Applications of IoT

There are huge challenges for research and development aimed to create a "smart" world linking the real, digital and virtual reality. IoT aims to allow people and physical objects to be connected anytime, anywhere, with anything and anyone [75]. It is expected that the raised value of information created by the multitude of IoT devices will be converted into knowledge for the benefit of the entire human-kind. IoT innovation is growing quickly and numerous applications [68] have been constructed so far, e.g. in the fields of wireless sensor networks (WSN), radio-frequency identification (RFID), information processing (using fog and cloud computing), security, intelligent homes, intelligent spaces and so forth. According to [76], we can currently divide the application sectors into six fundamental categories:

1) *Smart buildings* – including intelligent spaces (see chapter 5.2), where IoT technologies can reduce the resource consumption associated with operation of buildings, such as water, heating and electricity, as well as the improvement of people's satisfaction at work or home environment. In this category, the sensors are used to monitor resource consumption and actively detect and predict the needs of both users and other objects within the building.

However, such approach requires a high degree of object and protocol standardization in order to enable general adaptation of the technology [76].

2) *Smart cities* – require deployment of advanced communication infrastructure which would, for example, enable the optimization of road network utilization and therefore, increase the quality of living. IoT technologies can be used to direct the traffic with help of sensors and corresponding services delivering important information to commuters in order to avoid traffic jams. The information can be gathered from various types of sensors delivering all sort of data such as number of vehicles, average speed of vehicles or overall congestion. Additionally, smart parking systems will be able to monitor available free parking spaces and provide advice to drivers. City sensors can detect the level of air pollution, waste levels, etc. Early example of such information can be already seen on services such as Google Maps (see Fig. 40), however this is currently based mostly on smartphone data since vehicle sensory equipment and internet connectivity is not so well established yet [76].

Fig. 40 Traffic congestion overlay in Google Maps.

3) *Monitoring of the environment* – IoT technologies applied to the environment can collect information on temperature, wind, rainfall or waterfall level. By being interconnected they become capable of detecting and monitoring anomalies which could potentially threaten human life. Another useful application of IoT technology is represented by fire detection sensors, making fire departments informed about possible hazards in a very short time, that results in the rescue of human lives, the alleviation of property damage and the reduction of the impact of natural disasters in general [76].

4) *Healthcare* – patients will wear sensors to monitor their vital functions (e.g. temperature, blood pressure, pulse rate). Other sensors will monitor the physical activity of patients (using a gyroscopes and accelerometers). This information will be locally collected and sent to remote medical centers that will be able to respond quickly to changes as needed.
5) *Smart business* – where IoT technologies can identify and provide information about the movement of goods. RFID tags can be directly connected to shipping containers, allowing them to be tracked. They can monitor product availability in real time. Bio sensors can monitor food characteristics during transportation, such as temperature or number of bacteria in the consignment, in order to improve product quality and overall customer satisfaction. This sector of IoT application is growing rapidly with establishment of Industry 4.0 paradigm [76].
6) *Security* – security surveillance becomes a necessity for business buildings, shopping centers, factories and other places. It will provide a cheaper alternative to camera systems. Ambient sensors can be used to monitor the presence of hazardous chemicals. Human behavior sensors can be used to detect suspicious activity [76].

5.2. Intelligent Space

An *Intelligence Space* (IS) is a specific manifestation of IoT in which "*the actions of numerous networked controllers controlling different aspects of a common environment are orchestrated by pre-programmed pre-emptive processes (e.g., intelligent software agents) in such a way that it enhances experiences and capabilities of its occupants,*" either humans or machines [67].

The term often describes a place with considerable distribution of sensors and intelligent devices which are interconnected via communicating technologies often using the IoT paradigm. Such spaces may include a room, a corridor, a street or even entire city equipped with sensors and actuators. The goal of sensors is to acquire data about actions in the surrounding environment. Actuators use these data in order to interact with users of the space, usually humans and robots [72].

5.2.1. IoT Devices in Intelligent Space

The devices utilized in IS can be divided into four *fundamental categories* [75]:

- *sensors*,
- *actuators*,
- *interfaces*,
- *robots*.

One of the most frequently utilized *sensors* in IS are *cameras*. In combination with neural classification and identification systems, the IS can recognize common objects or people and propose corresponding actions. Since camera processing may be unreliable and depends on proper light conditions, it is common to utilize various other sensors for better precision. Examples of such are distance sensors, stereoscopic cameras, lasers or combined sensors such as Kinect. Other ways to identify objects is to employ RFID cards and readers, or Bluetooth Low Energy devices (such as E-Beacons) [75].

Another common type of device utilized in intelligent spaces is an *actuator*, which operates in the reverse direction of a sensor. While a sensor (such as camera or microphone) is used to transduce some form of energy or physical phenomenon into electrical impulse, an actuator does the opposite (see Fig. 41). It converts an electrical signal into a physical action. There are various types of actuators, such as automatic switches, electric motors, hydraulic systems, pneumatic systems and many other [75].

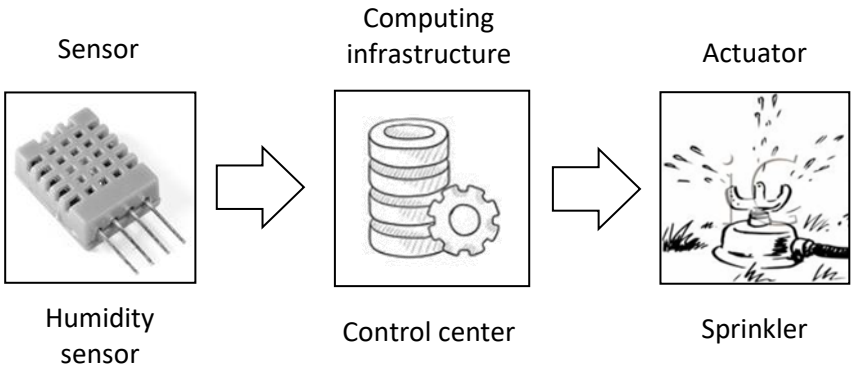


Fig. 41 Sensor and actuator cooperation.

Mobile *robots* can be used as intelligent agents that serve and interact with human users. A traditional concept of a mobile robot (as a *combination of sensors and actuators* within a single autonomous unit) is currently being replaced with an idea of *ubiquitous robotics* [88][89]. According to this paradigm, mobile robots operate using data provided from external sensors located within the IS and perform tasks correspondingly to the goals of the IS. An access to this additional information and processing capabilities of the IS increases an operation capacity of a robot. Robots are now able to move on from simple tasks towards more complex missions and reduce their dependency on human intervention.

Human intervention and direct control of various IS devices is performed via *human-computer user interfaces* (UIs). These interfaces are often provided in a form of mobile or web applications accessible from PCs, tablets and smartphones. The UI of application usually shows an overview of

the space using comprehensive dashboards (such as the IS monitor shown on Fig. 43). One of the most common applications of UIs is remote robot control, often described as teleoperation or telerobotics. The concept of telerobotics concerns with the control of robots from a distance, using network connection such as Wi-Fi, Bluetooth or other. It is often realized as *direct human-to-machine control* using UI. However, autonomous modes of *machine-to-machine control* using *application programming interfaces* (APIs) are becoming even more common. More on this topic may be found in chapter 6.3.

5.2.2. Example Proposal for Intelligent Space

As an example of Intelligent Space, we propose an ongoing implementation [71] executed at the Center for Intelligent Technologies (CIT) [87]. Floor plan of this space is show in Fig. 42.

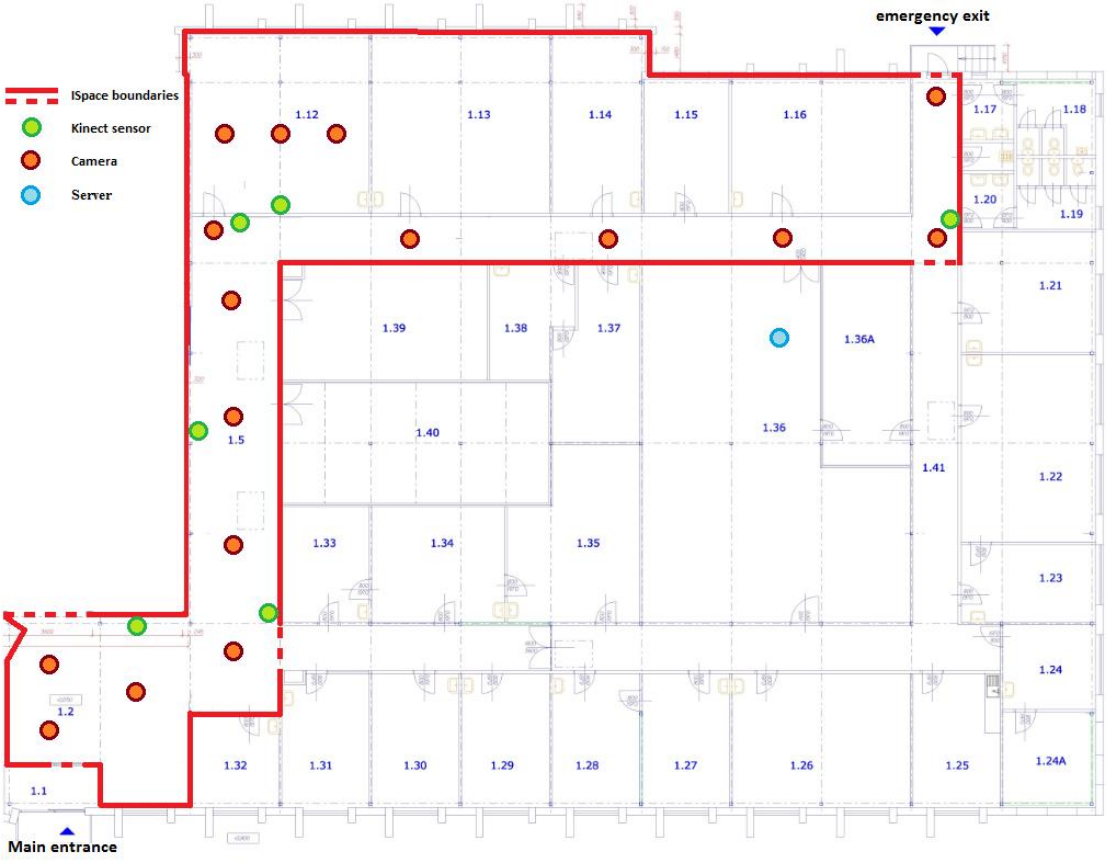


Fig. 42 Floor plan of the Intelligent Space at CIT [88].

The space includes several types of sensors including Kinect sensors, IP cameras and microphones, ambient sensors for light intensity, temperature, air humidity and more. Fig. 43 displays an overview of the space using the camera monitoring web application. In addition to conventional Ethernet and Wi-Fi enabled devices, the implementation expects utilization of other technologies, most notably the Bluetooth Low Energy E-Beacons and RFID [88].

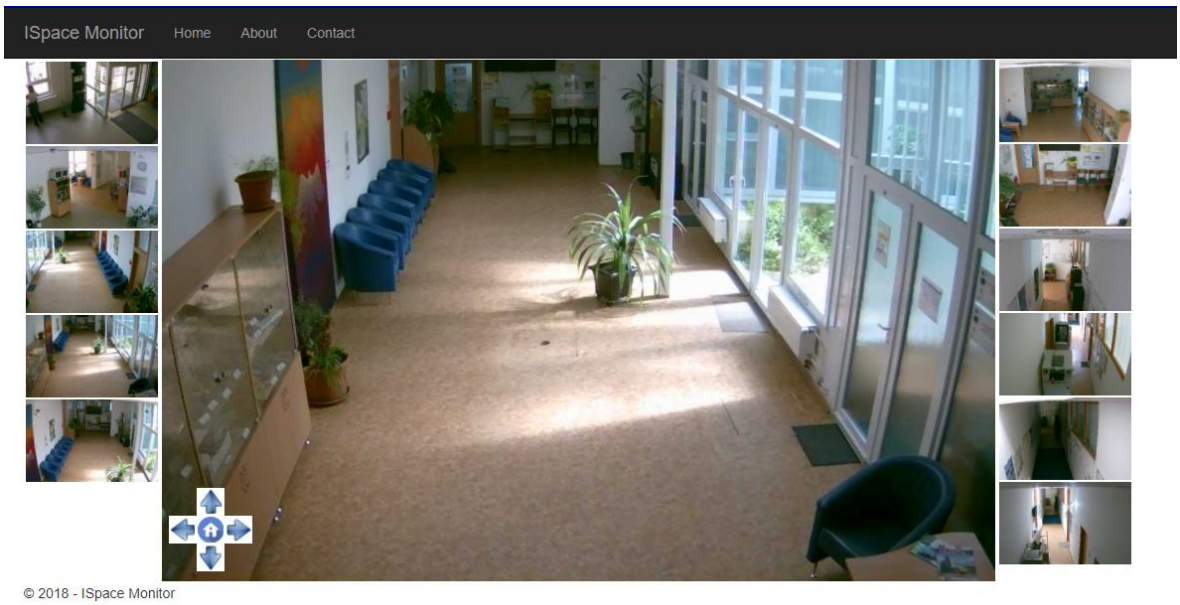


Fig. 43 Monitoring system of the Intelligent Space at CIT [88].

With regard to utilized robotic hardware, the space is equipped with several types of mobile robots (see Fig. 44) including Aldebaran Nao, SONY AIBO, TheCorpora Qbo, Robotnik TurtleBot and others. Control of these robots is performed by Robot Operating System (ROS) which enables universal remote control of various types of robots and is well applicable to the IoT paradigm [71].

Fig. 44 Robotic hardware utilized in IS at CIT [71].

The networking proposal [71] (displayed on Fig. 45) clearly shows, that the Intelligent Space records various data, such as video streams from IP cameras, depth maps from Kinects, sound from microphones and other data from ambient IoT sensors. Estimated dataflow from the sensors is displayed in Tab. 8. The entire Intelligent Space is expected to produce data at a rate of at least 260 Mb per second or 21,5 Tb per day, which is a considerable amount.

Tab. 8 Estimated dataflow from various sensors in IS at CIT [71].

Sensor type	Data description	Count	Estimated data flow per second	Estimated amount of data stored per day
IP Camera	MJPEG stream HD video at 24 FPS	16	16×10 Mb/s	16×850 Gb
Kinect	HD video at 24FPS + depth map + skeleton contour overlay	6	6×16 Mb/s	6×1350 Gb
Microphone	24-bit/192kHz single channel audio	16	16×256 kb/s	16×21 Gb
Ambient sensor	the value of an arbitrary physical quantity	varied	*n*×64 b/s	*n*×5 Mb
Toggle sensor	externally triggered event timestamp	varied	varied	*n*×1 Mb
		Total:	**~ 261 Mb/s**	**~ 21,5 Tb**

In order to transmit, process and store this amount of data, it was necessary to define viable strategies. Estimated dataflow naturally excludes a direct use of cloud services, especially for video processing, which needs to be performed on local servers using fog computing infrastructure (including acquisition, processing and storage). However, audio and other data can be processed directly via cloud services since these are not so bandwidth demanding [71].

Therefore, the proposal [71] assumes to use two local networks bridged via fog computing servers. The first, hidden network (LAN1) connects the bandwidth-demanding cameras and Kinects directly to the servers via high capacity 1 Gb/s optical connection. The second, publicly open network (LAN2) connects other devices, including low bandwidth wireless IoT sensors, robots, users' PCs and other clients. The access to video streams from the cameras and the Kinect sensors (connected to LAN1) is possible via an interfacing application deployed on the fog computing servers [71].

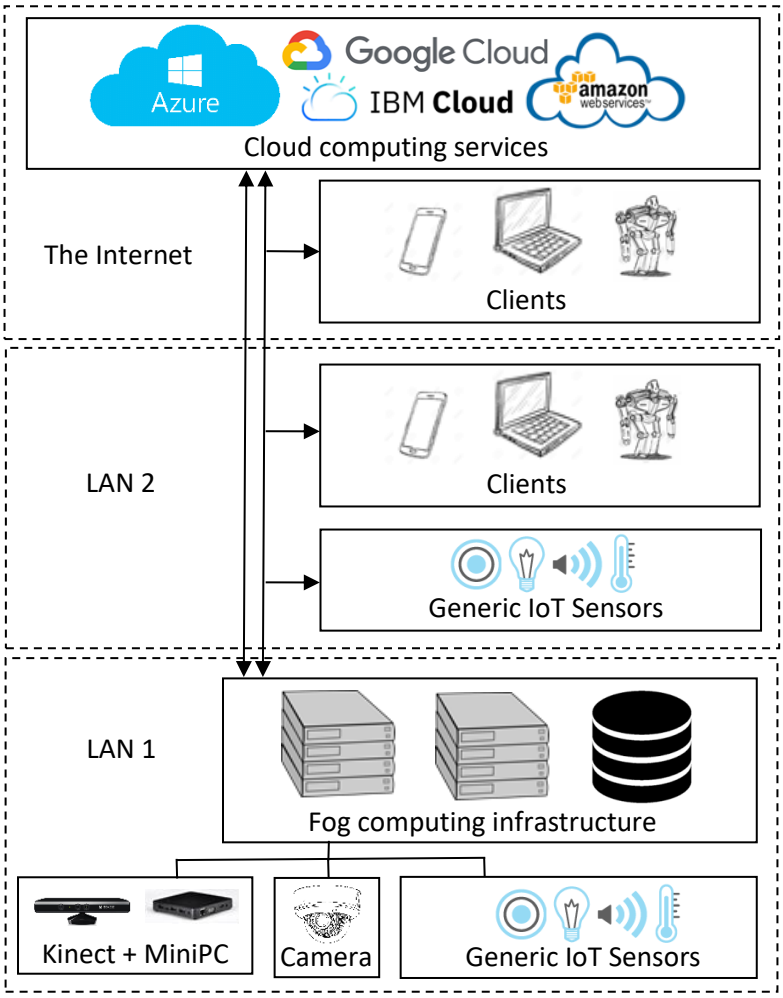


Fig. 45 Networking Proposal of IS at CIT.

5.2.3. Applications in Intelligent Space

The number of possible *use cases* of intelligent space is only limited by our imagination and technical capabilities. In this section, we present some applications which were already implemented and tested at CIT along with ideas from other research collectives and businesses.

There are several IoT applications focused on *home automation and monitoring*. Various ambient physical values, such as humidity, CO2 concentration or lighting conditions, can be monitored and controlled but the most common use case is an *automated temperature control* [93]. Intelligent space paradigm can be applied to households in order to effectively control several heating actuators using the information from array of sensors including simple thermometers, but also cameras and other personal detection sensors, which may be combined with prediction algorithms running within an internal fog computing infrastructure. Using such an approach, the resources used for heating can be optimized, for instance by reducing the heating performance in case of absence of household inhabitants. A virtual model of such household can be made with precise

temporal and physical properties including external weather impact, further improving the effectiveness of resource consumption [93].

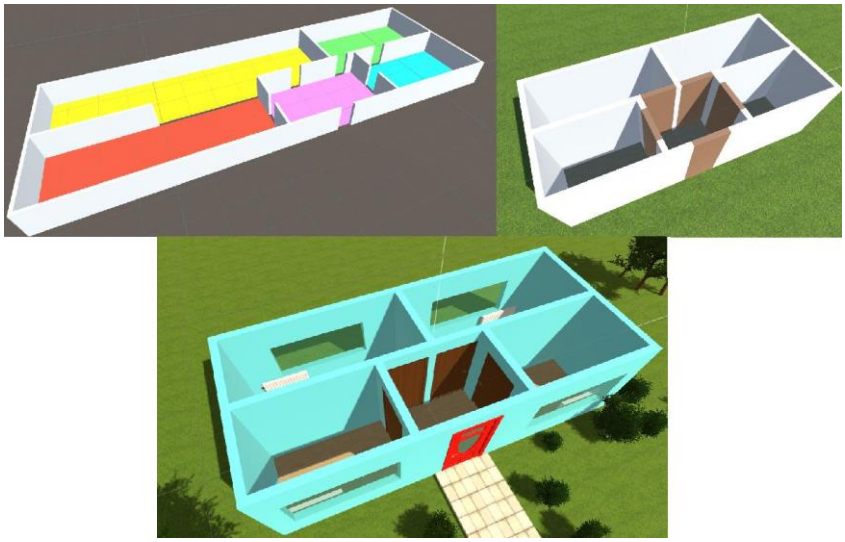


Fig. 46 Virtual model of an intelligent household [93].

Video monitoring using CCTV or *IP cameras* and similar optical sensors is being extensively applied as well [94][95]. Cameras are usually employed for surveillance of the space and event detection. Video streams may be directly overviewed by a security officer or recorded and analyzed later, only if necessary. Intelligent space applications often use automatic event detection [94]. Typical events include:

- simple movement detection,
- object detection, localization and identification,
- face recognition, person identification and localization.

The events may be analyzed and used by other agents (e.g. robots) within the space immediately as they occur or stored for later processing and data mining. All of the events can be marked with a timestamp and stored to a database. Since images are archived only if detection is made, required storage capacity is significantly lower compared to continual recording and available computing resources are being conserved. Additional information and more precise localization of events can be obtained by employing more advanced sensors such as Kinect [95].

Another interesting range of applications deal with *sound processing* and its analysis, speech recognition, speech-to-text transformation and emotion detection [92][96]. The main task is to adequately solve these issues and programmatically implement them within an intelligent space, which would work autonomously with satisfactory reliability. Available solutions often utilize cloud services for speech-to-text processing as well as emotion analysis (see Fig. 47).

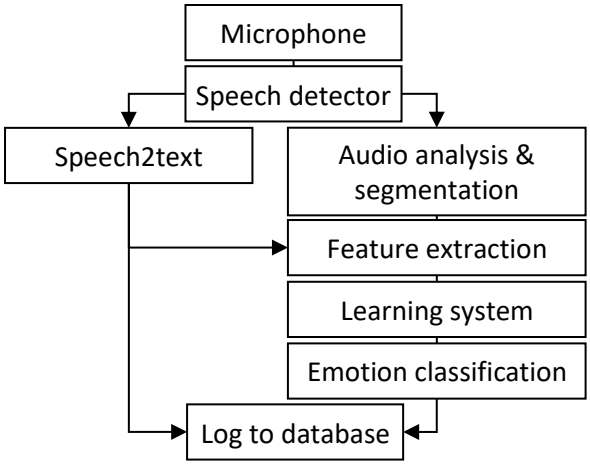


Fig. 47 Proposal of speech emotion recognition & text transcription system [71].

The results of speech recognition and emotion analysis may be thereafter used as a source of additional *events* occurring within the intelligent space which can be reacted upon by various actuators (speakers, thematic lighting) and autonomous agents (robots) in order to fulfil the needs of human users. In addition, speech recognition systems may be applied as complementary human-computer interface since voice commands provide users with an alternative to the conventional graphical control applications (web applications, smartphone applications or stationary wall mounted terminals).

Both video and audio analysis provided by intelligent space can be used for automatic personal identification and authentication in *attendance systems* or *visitor systems* [97]. Visitor systems increase overall work performance of receptionists by allowing them to work on other important projects while visitors sign themselves in. Using cameras and facial recognition, this process can be fully automatic, convenient and inconspicuous. It also enhances the safety and security of a facility while ensuring visitor comfort. Each detection event can be stored within database and further processed by attendance system in order to generate accurate working times which may be used to automatically generate payrolls and other necessary documentation.

An interesting synergy of ambient sensors for *environment monitoring* and user *localization services* is shown in [102] (see Fig. 48). The proposed solution is composed of a mobile robotic platform and a smart environment in two remote experimental sites integrated through a cloud platform, specifically a domotic flat located in Italy (Domocasa Lab, Peccioli, IT), and an assisted residential condominium in Sweden (Angen site, Orebro, SE).

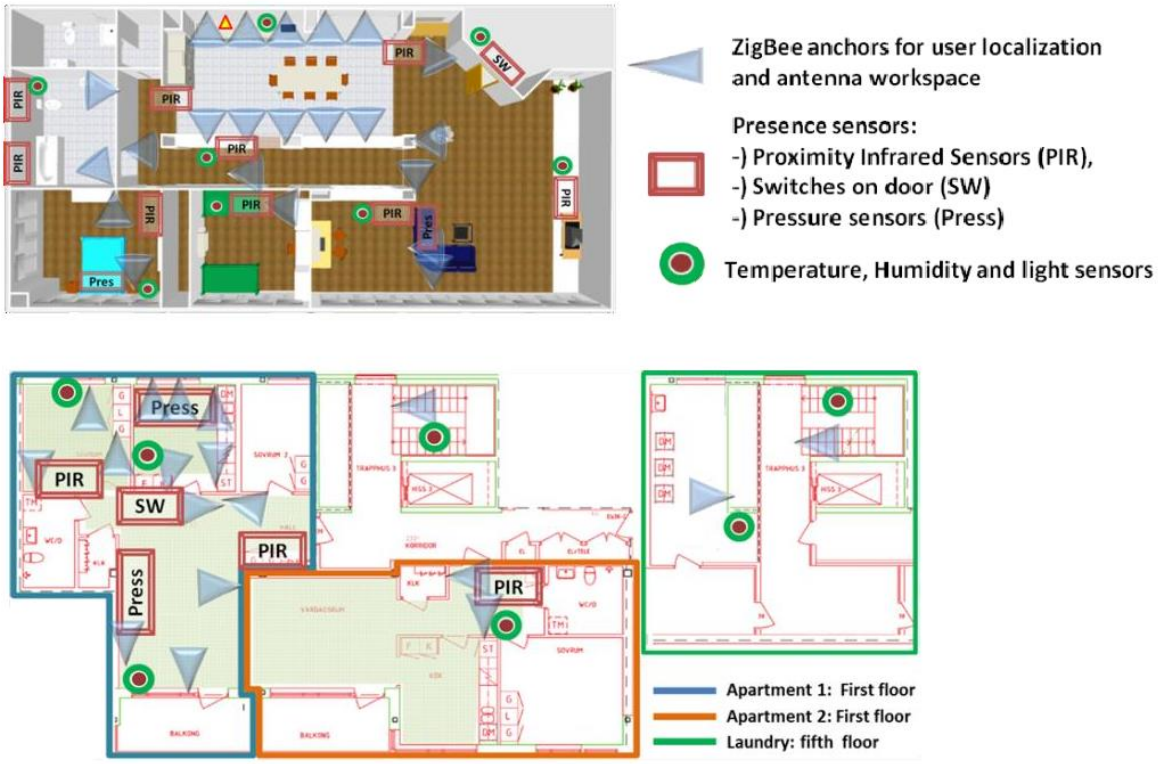


Fig. 48 Description of intelligent spaces [102].

The goal of the implementation is to provide a platform where one caregiver is able to help more elderly people at the same time, regardless of time or location of seniors and the number of services to be addressed simultaneously. The implemented features include an *indoor personal localization* via heterogenous sensors (including wearable Zigbee locators, proximity sensors and other ad-hoc sensors), *notification service* (which reminds users about their appointment or medication) and environmental *monitoring service* (overviewing the home status and alerting users and caregivers in case of critical situations) [102].

Another perspective view on the *future of home automation* was recently presented by a Facebook CEO in his article on the social network [103]. His goal was to build a rudimentary AI, nicknamed *Jarvis*, capable of automatically controlling various parts of home including heating, lighting, music, appliances, security and more, while being able to interact with the members of the household via voice and text interfaces (see Fig. 49). The system learns behavioral patterns of household users, can learn new words and concepts, and proposes automation of most commonly repeated tasks. It uses several artificial intelligence algorithms, including natural language processing, speech recognition, face recognition, and reinforcement learning, written in several programming languages [103].

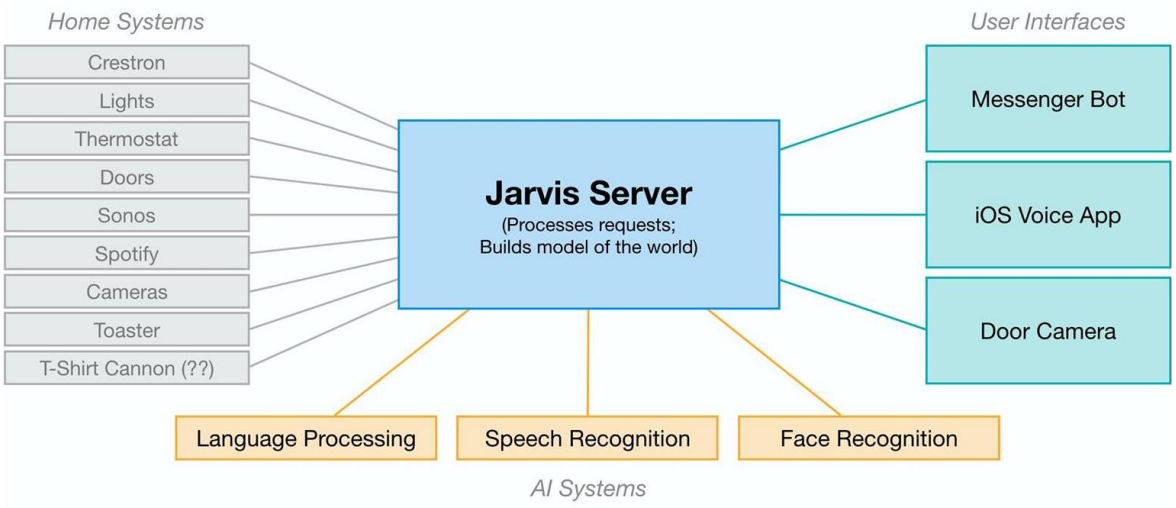


Fig. 49 Diagram of the home automation system Jarvis [103].

The article [103] includes an important commentary about mutual compatibility of various commercial automation systems (lighting, thermostats, security doors etc.) and online services. The interconnection of these systems is difficult, since they use different programming languages and communication protocols. Often, it is necessary to *reverse engineer APIs* for some of these in order to issue a simple command to turn the lights on or get a song to play. In addition, most appliances aren't even connected to the internet yet and it is necessary control them using internet-connected power switches that turn the power on and off remotely. But there are still some devices (e.g. toasters), which cannot be controlled using such approach. The commentary concludes, that in order to make home automation accessible for general public, it is necessary to make more devices connected to the network. Finally, the industry needs to develop *common APIs* and standard protocols for communication between devices [103].

5.2.4. Ubiquitous Robotics in Intelligent Space

Service robots represent another promising feature of intelligent spaces [88][98][99]. However, it is a well-known fact, that fully autonomous control of robots is still considered a challenging task, which necessarily deals with various problems such as robot localization, object tracking, path planning, navigation, movement control, obstacle detection, avoidance etc. Since these problems are difficult to be solved by a single autonomous unit, the attention of researchers and businesses is being focused towards so-called *ubiquitous robotics* [88][90], which consider a robot in an intelligent space as an agent consisting of three separate components [100]:

- *real mobile robot*,
- *virtual robot* model (avatar),
- *separated sensory system* in a surrounding environment.

This combination of independent components enables a robot to perform more complex tasks than a traditional autonomous unit, thanks to the use of additional sensors located within intelligent space and additional computing capabilities provided by either fog or cloud computing infrastructure. Virtual models (or controllers) can be trained and improved independently from the physical robot and switched (see Chapter 7.1) according to the specific situation (in line with the methodology of Situational Control, presented in Chapter 1.4.1). There are many possible *use cases* for service robots [88][90][98][99], for example:

- human *escort* or *guide*,
- *receptionist* or *secretary* (see Fig. 50),
- *messenger* or *courier*,
- *security officer*,
- *cleaner or janitor*,
- and many other.

It is obvious that a mobile robot is a suitable addition to any intelligent space, since it features an array of sensors, actuators and human-computer interfaces, all conveniently included inside a single body. With the introduction of a ubiquitous robotics paradigm, pervasive IoT technologies and recent progress in AI methods, serious employment of robots in previously mentioned tasks becomes a real possibility in a very near future.

Fig. 50 Visit notification using facial recognition via Pepper robot [101].

5.3. Intelligent Space as Complex System

One of the **scientific contributions** of this dissertation is "*the proposal of a situational control model of an intelligent space utilizing fuzzy cognitive maps".* Since we aim to apply a methodology of *situational control* (see chapter 1.4) intended to be used for management and control of *complex systems*, we would like to justify its application by an unambiguous confirmation of the *intelligent space* as a *complex system*.

In general, a *complex system* can be described as such a system, where there is a large number of forming elements, which total quantity, mutual relations and organization structure can be changed at any time. The elements form a hierarchical structure while retaining the right of autonomous decision-making [4].

In comparison, the *internet* is probably the most complex system humans have ever designed. It forms the hierarchical structure as the network of networks, where each network utilizes similar globally recognized technologies and protocols. Each network device cooperates according to the ISO/OSI model, which is hierarchical by its definition.

Since the *intelligent space* is based on paradigms of the global internet and IoT, it can be assumed as complex system (as it is demonstrated in Tab. 9) and therefore, all the management and control methodologies (proposed in chapter 1.3) may be applied.

A proper *control of intelligent space* naturally requires appropriate software tools, powerful enough to fit various use cases. For this purpose, we propose a novel *fuzzy cognitive maps library* and corresponding *application programming interface*, described in the following parts of this work.

Tab. 9 Intelligent Space compared to properties of Complex System (according to chapter 1.1)

Complex System (CS)	**Intelligent Space (IS)**
• a *large number of elements* which number can increase or decrease,	• a *large number of IoT devices* which can be added or removed accordingly,
• *relationships between the elements* defining the structure of the CS may change,	• *mutual communication between IoT devices* may change according to the specific use case of the IS,
• *hierarchy of the organizational structure* of the components of the CS (regardless of whether it is a management or controlled system),	• *hierarchy of networking components* of the IS corresponding to different layers of ISO/OSI model, hierarchical utilization of cloud/fog/edge computing paradigms,
• *autonomous decision-making of elements* (decentralized organization of the CS, where individual elements have large degree of autonomy),	• *independent IoT devices* (decentralized sensors, actuators and other devices, integration of edge computing into devices),
• *change of space* (dimension of space) in which the CS is located and in which it operates,	• *change of space disposition* (objects within the space), relocation of sensors, actuators and other equipment,
• *change in the organizational structure* of the CS (i.e. changing links or relationships between elements of the CS).	• *change of networking technologies and communication protocols*, reassignment of devices to other subnets.

6. Software Tools for Control of Intelligent Space using FCM

"People often think that computer science is the art of geniuses but the actual reality is the opposite, just many people doing things that build on each other, like a wall of mini stones."

Donald Knuth

In this chapter we present an overview of dedicated software tools, specifically suitable to be used for FCM applications, together with their advantages and disadvantages. After that, we introduce a new multi-purpose FCM library along with its preliminary requirements and a couple of potential proposals and implementations. The introduced library tries to provide flexibility and applicability for various use cases, from simple system prototyping to modeling and control of complex systems, including wide IoT networks and intelligent spaces. In contrast to the most of the available FCM tools, the library is also focused on integration of a built-in learning mechanism for automated adaptation of the parameters of the map [104].

This chapter therefore presents the ***second contribution*** of this dissertation (*" the creation of a programming library for FCM computations applicable to modeling and control of complex systems, including auxiliary graphical user interface (GUI) and application programming interface (API)"*).

6.1. Overview of Existing FCM Tools and Frameworks

There are several tools and libraries available for analysis and system modeling using FCMs [104]. Examples of these tools are:

- FCMapper [105],
- Fuzzy Cognitive Maps Applet [106],
- Java Fuzzy Cognitive Maps library [107],
- Mental Modeler [108],
- FCM Wizard [109],
- extensions for MATLAB/Simulink (e.g. FCM Tool [111] or Fuzzy Logic Toolbox [112]).

All of these tools have advantages as well as some disadvantages, which were thoroughly discussed in [104]. In a next few sections we present a short excerpt of this discussion.

6.1.1. Brief Description of Available FCM Tools and Frameworks

FCMapper [105] is an FCM extension for MS Office Excel spreadsheet processor and Visual Basic for Applications (VBA) framework. It provides means for basic system analysis and modeling. The visualization of FCMs is not directly supported, but created models can be exported and visualized in external visualization programs [104].

Fuzzy Cognitive Maps Applet [106] is a Java applet application embeddable within a webpage. It requires Java Runtime Environment (JRE) or an appropriate web-browser plugin in order to run. There is no need for applet installation, it automatically loaded along with a webpage. However, since it is not possible to save created models (as a security feature of all the applets), the tool is mostly suited for educational purposes or simple FCM prototyping [104].

Mental Modeler [108] is FCM modeling software which can be used in two ways, either as a desktop application or as an online tool. It provides a graphical user interface which allows easy and intuitive building of FCMs even for larger groups of users who cooperate on model creation. The tool is expert oriented and does not provide automated learning options [104].

FCM Wizard [109] is another recently published web-based modeling and analytics tool. There is not much information available, since the tool is proprietary and closed, being available only to a small group of researchers and workshop [110] attendants.

Java Fuzzy Cognitive Maps library (JFCM) [107] is a library written in Java, which provides classes and methods usable in another programs. Since it is provided as open source, it can be easily extended if built-in components are not sufficient. Being a Java library, it requires the JRE in order to run [104].

FCM Tool [111] is an embedded application executed from MATLAB command line and running in a separate window. It enables users to work with FCMs in a form of matrices and concept values during simulation can be visualized on 2D plot. It does not provide any tools for automatic adaptation of FCM relations, manual setup is necessary [104].

Fuzzy Logic Toolbox [112] is an official extension of MATLAB. It provides additional functions, applications and a Simulink blocks for analyzing, designing, and simulating systems based on fuzzy logic, including automatic adaptation by fuzzy clustering or adaptive neuro-fuzzy learning. Applications include the *Fuzzy Logic Designer* and the *Neuro-Fuzzy Designer*, both of which are provided with a convenient graphical user interface. Fuzzy *blocks in Simulink* (see Fig. 51) are an alternative to the previously mentioned programs with a graphical user interface [104].

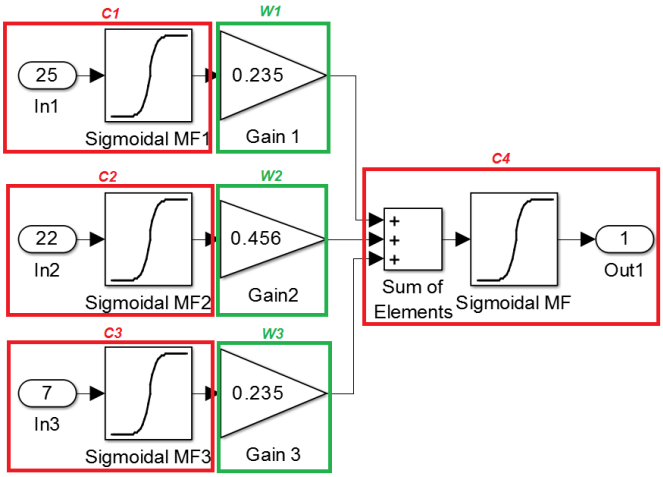


Fig. 51 MATLAB/Simulink model of a simple FCM with three concepts connected to the fourth concept where weights are represented as gains. Values from the real world (inputs) are fuzzified using the sigmoidal membership function. The sigmoidal function is also used as a threshold function for an activation of the fourth concept [104].

6.1.2. Categorization of Available FCM Tools

According to [104], the tools introduced in the previous chapter can be incorporated to the following categories with regard to their utilization potential:

- *Single-purpose applications*, such as Fuzzy Cognitive Maps Applet, are limited to trivial tasks, such as prototyping or demonstration of basic principles of FCM in education. Advanced functions, such as import or export of created models are not available.
- *Limited multi-purpose applications*, such as FCMapper enable wider utilization range, possibility to export created models and its use with other programs. This class of applications is often embedded in superior program (e.g. spreadsheet processor) and generally simple to use. However, its functionality is hardly extended.
- *Extendable multi-purpose applications*, such as Fuzzy Logic Toolbox or FCM Tool, provide the widest range of options and possibilities. Such applications can be extended and interconnected with other applications and frameworks, allowing the control of real-world dynamic systems.
- *Multi-purpose libraries*, such as the JFCM, whose design is deliberately oriented to cover as large set of possible applications as possible. While these libraries usually do not include the user interfaces, they provide a large set of simply deployable functions which cover the fundamental background logic. These functions can be used to rapidly prototype any kind of necessary application.

6.1.3. Deficiencies of Available FCM Tools

The overview in [104] further discusses the problems and shortcomings of the described tools. These are often platform or implementation related, and the most relevant can be summarized as follows:

- *Restrictions of embedded applications* (MS Office spreadsheets, Java applets, etc.) due to limited capabilities of a parent program (spreadsheet processor, web browser). This often affects basic functionalities, such as import/export of FCM models or model visualization.
- *Dependency on external software*, often proprietary (MS Office, Java, MATLAB, etc.), which limits the accessibility and deployment options of dependent tools. These limitations do not directly affect the application usage or model implementation. The constraints are mostly economical, which may not be an issue in some cases. However, widely introduced licensing (e.g. in larger education facilities, or in distribution for general public) is expensive for some software (such as MATLAB).
- *Dependency on external frameworks and libraries* (JRE, MATLAB Runtime Compiler, .NET, etc.), where even standalone FCM related programs require the installation of additional runtime components. These are often proprietary and not available for all OS platforms.

6.2. Open Fuzzy Cognitive Maps Library

Even though we have introduced several existing tools usable for FCM modeling and control, unfortunately almost none of them can be considered complete and applicable in all situations. A practical application of most of these tools is severely limited and even the most universal ones, such as JFCM library, does not support advanced FCM designs necessary to control complex systems (see chapter 4.2). Further, most of the tools lack even the most basic support for automated model adaptation and learning (see chapter 4.3).

With all of this in mind, we created a novel FCM library [104] thoroughly described in this chapter, including its preliminary requirements [104], an initial proposal and implementation [114] and a final production-ready version [115].

6.2.1. Library Requirements

Considering all of the relevant features and deficiencies of FCM tools mentioned in chapter 6.1, we created a list [104] of basic requirements for a new FCM library. According to the list [104], the library needs to provide the following properties and features:

- *Map design*, including commands for definition of concepts and commands for mutual connections of concepts.
- *Setup membership functions*, i.e. define *fuzzification* and *defuzzification* of concept values (transformation from real-world values to fuzzy values and vice-versa).
- *Definition of casual relations between concepts*, including nonlinear and dynamic relations (with regard to chapter 4.2).
- *Concept initialization* via commands capable to manually setup concept activation values (in case of use of input nodes with controllers and sensors [40]).
- *Update of FCM* during a runtime. The updated activation values of map concepts need to be computed in a reasonable time. Therefore, it is beneficial to support parallelized and scalable algorithms.
- *Automated FCM adjustment* of relations using basic learning methods. Other specialized methods should be easily implemented via appropriate interface.
- *Model import/export* (i.e. file input/output), preferably in standardized format compatible with other FCM libraries and applications.

In addition to these basic features, the library should be easily deployed on various platforms and operating systems. It should use as few third-party dependencies as possible and be independent from expensive or proprietary runtime frameworks (such as MRC). Preferably, the library should use an open source licensing in distribution to enable simple updates depending on additional requirements of a specific application or a user [104].

6.2.2. Preliminary Library Proposal

Our initial proposal [104] for the new multi-purpose FCM library was based upon the assumptions introduced in the previous chapter. A system diagram of the proposal is displayed in Fig. 52. A *high-level interaction* with the library is done via FCM object, which attributes can be set directly by an expert using the proposed *design module* commands, such as [104]:

- addition or removal of concepts,
- renaming and initialization of concepts,
- setup of membership functions,
- declaring relations between concepts,
- connecting concepts.

The proposal assumes, that relations between concepts can be also defined by an expert through a *low-level interaction* commands and defined either as [104]:

- simple linear weights,
- predefined nonlinear analytic functions,
- or TTR-NFCM (see chapter 4.2.3).

Therefore, by using the design module, it is possible to create a map entirely with use of an expert knowledge. However, the proposal also expects a *learning module* included within the library, which could be used to adjust some parameters automatically, eliminating the need to use manual *low-level interaction* commands.

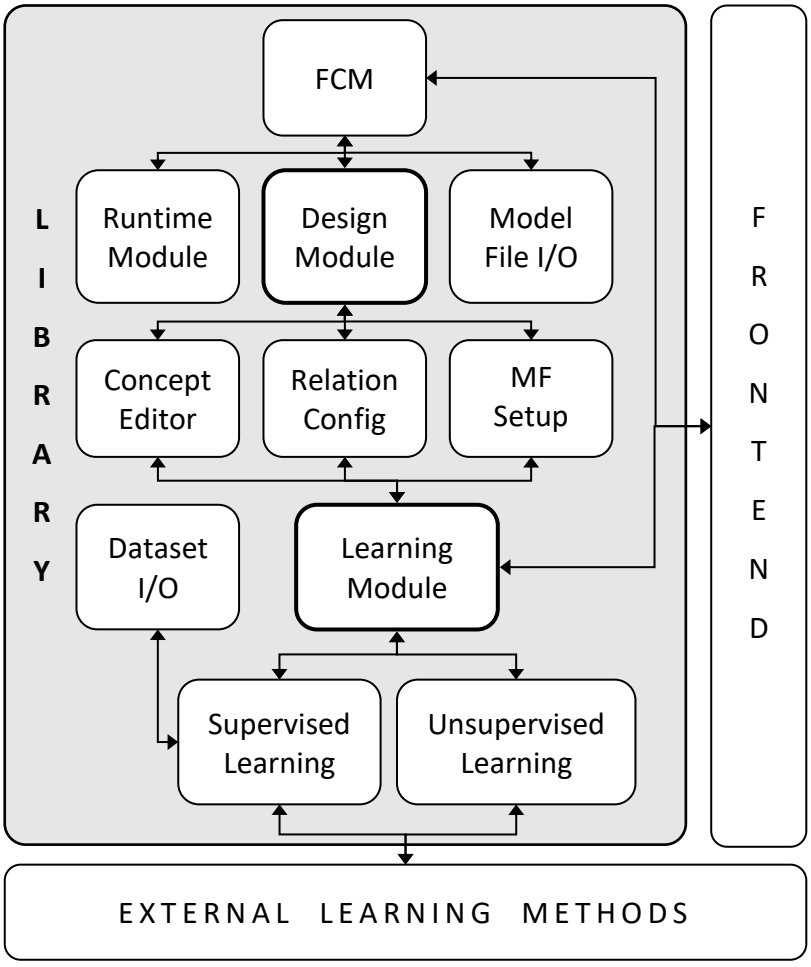


Fig. 52 System diagram of the preliminary proposal for FCM library [104].

The proposal [104] expected an integration of both supervised and unsupervised learning methods, which could be applied in order to adjust either the relations between concepts or even generate entirely new concepts and relations. As built-in *unsupervised* learning methods we proposed mostly the family of *Hebbian based learning* methods. For *supervised* learning, we proposed both the *evolutionary or population-based optimization algorithms* and the *LMS optimization methods* from

area of neural networks. The proposal further expected a specialized interface which would enable simple addition of *external learning methods* via extendable plugins [104].

The user interface or *frontend* was not expected to be included directly in the library, but to be created as a standalone GUI application, a web application or a MATLAB/Simulink module.

6.2.3. Preliminary Implementation in .NET framework

The implementation related aspects were also considered during the selection of available frameworks and programming languages suitable for the development and deployment of the library. Therefore, in addition to the requirements from chapter 6.2.1, several additional recommendations were assumed, including [104]:

- avoidance of platforms which already have available FCM tools at disposal,
- availability of the library on all of the relevant operating systems,
- simplicity of development and implementation,
- ease of use, deployment and installation.

With respect to the first point, in spite of a lack of some features, we acknowledged JFCM library as a very suitable foundation for FCM community in Java environment. Since more competition in this area may not have been constructive, other options were considered. Initially, we aimed the implementation for a deployment on desktop operating systems which are mostly used for productive work, in which case the MS Windows operating system is the most prevalent. The possibilities of development and deployment were also in favor of this platform.

Accordingly, we decided to implement the library using the *.NET framework*, which is an integral part of widely available Windows platform. Additionally, since parts of the framework were recently released as open source and integrated into Mono [113], it is now possible to deploy the library implementation on Unix based operating systems (such as GNU/Linux and Mac OSX) [104]. Cloud based support of .NET was also considered favorable.

As a result, we have created an OpenFCM, a *.NET shared dynamic library* (DLL) [114], which could be easily incorporated into programs developed in other programming languages and environments including Python, MATLAB/Simulink, etc. The preliminary implementation included the following functions and commands for high-level interaction [116]:

- add, remove, connect and disconnect concepts,
- change concept relations (weights),
- list concept names, rename concepts,

- get and set concept activation value,
- set concept fuzzification functions,
- save and load FCM model.

The library repository [114] further included a simple testing application used to execute and debug individual library commands and display the results using Windows Presentation Foundation (WPF). The integration of DLL with MATLAB environment was also successful, with a possibility to access all the functions from MATLAB command line after a single one-line import command [117].

Separately from the library we have also created an interactive GUI [118] (see Fig. 53) based on Windows Forms and Microsoft Automatic Graph Layout (MSAGL) [119]. The MSAGL library provided a support for online user interaction with the designed FCM and also allowed to render the updates of concept activation values in the real-time during map simulation. The GUI wrapped all the functions provided by FCM library DLL [118].

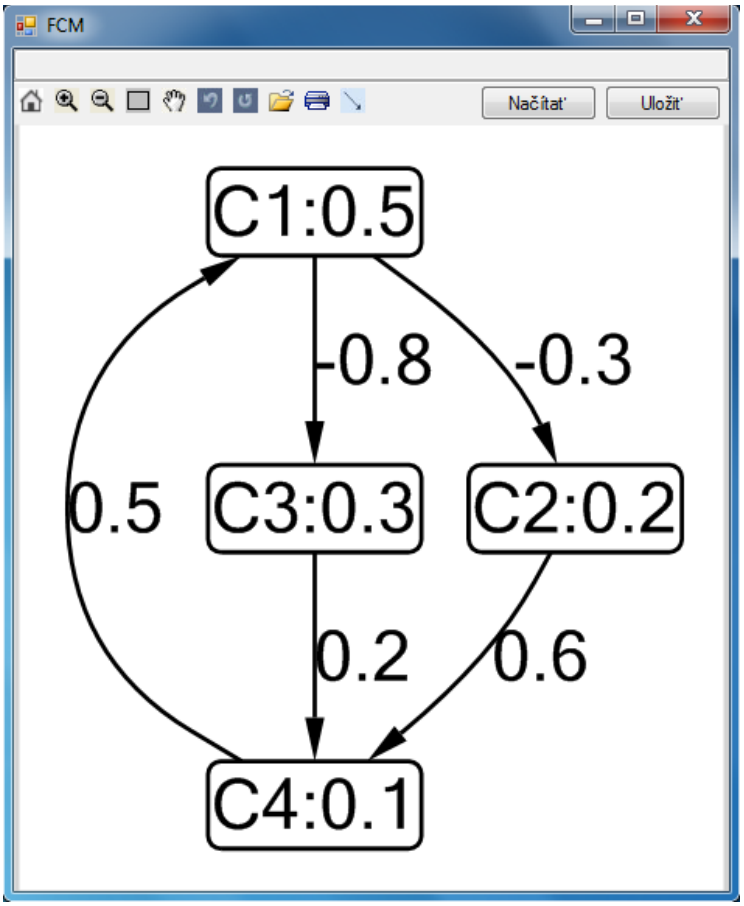


Fig. 53 Graphical user interface for OpenFCM Library [118].

However, even though all the *high-level* functions were implemented, both in the library and in its complementary GUI, there are several features included in the preliminary proposal (chapter 6.2.2) which failed to be implemented, most notably the *low-level* relation configuration and automated learning methods.

It became obvious, that the proposal was heavily focused on *high-level* interaction, which worked as expected. However, advanced FCM designs and automated learning were very dependent on the *low-level* functions which were often difficult to be implemented directly through the FCM object in its entirety.

In addition, a *platform-related problem* was discovered during the efforts to deploy the library directly on robotic hardware and in embedded applications, where .NET support was often not available. This was not so obvious a few years ago, however, with a rise of IoT applications it become a limiting factor. Therefore, a further change in the proposal and possible re-implementation of the library was considered appropriate.

6.2.4. Final Library Proposal

The preliminary proposal (chapter 6.2.2) for FCM library was heavily focused on high-level interactions with the library's FCM object through universal and simple commands. Such an approach was excellent for the creation of FCMs with simple linear relations and membership functions. However, with a need to use more complex relations such as TTR-NFCM and nonlinear membership functions a change in a design of the library was necessary.

The final proposal (see Fig. 54) has changed several aspects of the library. In addition to the main *FCM object*, which is still accessible via high-level commands, we introduced a direct access to the *Concept object*, which makes it possible to use additional low-level commands. The setup of membership functions was moved to a group of a new low-level commands available directly via the Concept object. Similarly, the setup of relations was enabled through the Concept object as well. With the final proposal, these basic operations are available via the FCM object:

- addition, removal, renaming and initialization of concepts,
- configuration of default relations and membership functions for new concepts,
- connecting and disconnecting concepts,
- import and export of FCM model,
- calculation of FCM updates during runtime.

Additionally, these *low-level* commands are available via the Concept object:

- setup and evaluation of input MF,
- setup and evaluation of output MF,
- setup and interaction with concept relations including:
    - attaching and detaching preceding concepts from a relation,
    - forward propagation of activations of preceding concepts through the relation,
    - backpropagation of error to preceding concepts,
    - adaptation of relation according to the error.

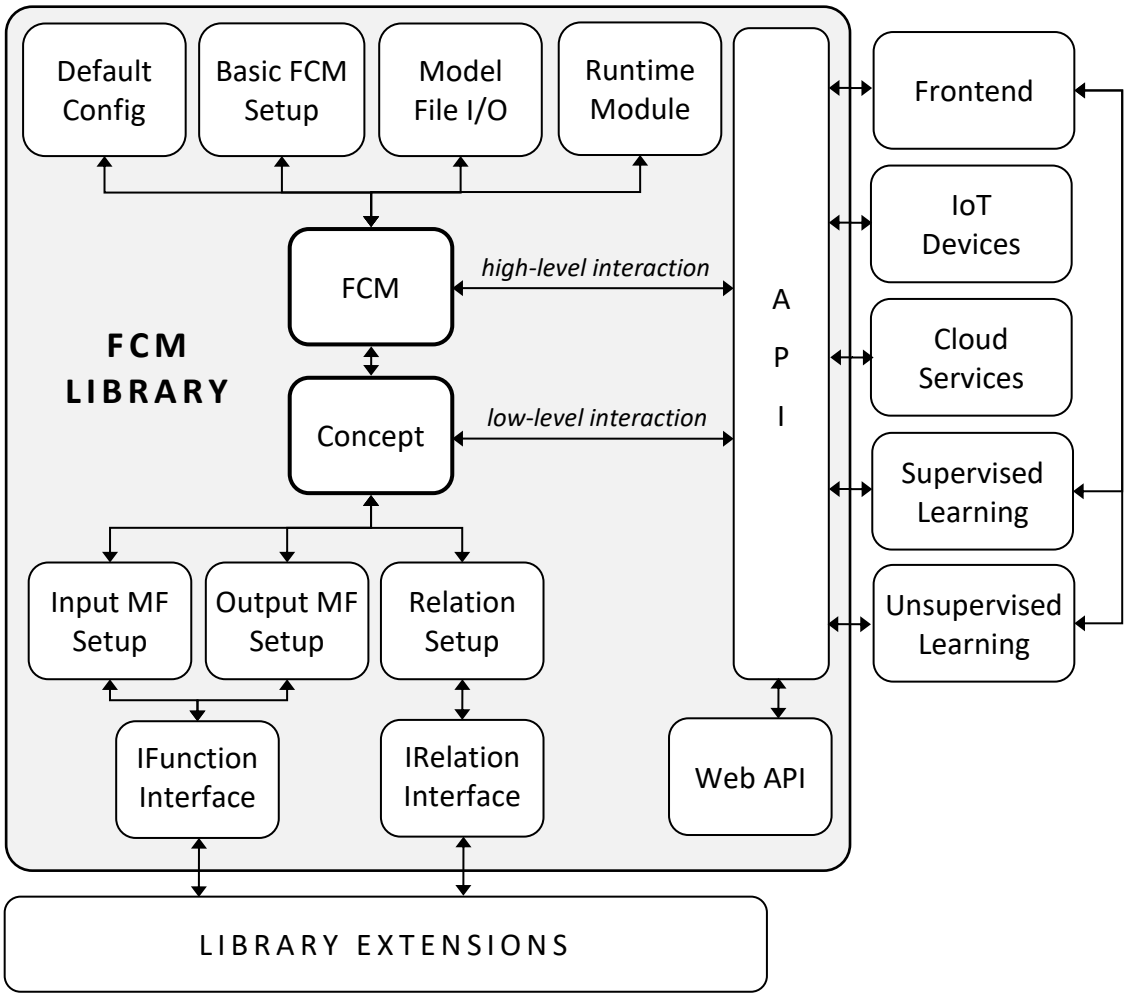


Fig. 54 System diagram of the final proposal for FCM library.

With regards to learning algorithms and adaptation of FCM, we recognized the duplicity in the preliminary proposal, where both the *design module* and the *learning module* needed to be adjusted in order to be compatible with every new type of relation. The final proposal aims to mitigate this duplicity by relaxing the library requirements. The final proposal therefore does not necessarily require the integration of all the learning methods directly into the library.

With respect to the similarity of FCMs with neural networks, only the support for supervised learning via least-mean-square methods and error backpropagation is proposed to be directly integrated into the implementation of each individual type of relation. According to the proposal, each concept can have expected values set-up which can be thereafter compared with calculated values and finally, the error can be backpropagated, if the implementation of selected relation supports it. The implemented backpropagation functionality can be called via respective low-level interaction commands.

Additional types of relations with a support for various LMS methods can be added to the library using predefined interface[1] *IRelation*, which requires the implementation of the specific methods including:

- get() – returns all the adaptable parameters of the relation,
- set(params) – sets the parameters of the relation to the specified values,
- attach(concept) – adds a new preceding concept to the relation,
- detach(concept) – disconnects the concept from the relation,
- propagate() – calculates a new value for the current concept,
- backprop(error) – backpropagates the error through the relation to preceding concepts,
- adapt(error, learning rate) – adapts the relation according to the error.

A similar interface *IFunction* is also specified for additionally defined membership functions. It requires the implementation of these methods:

- get() – returns all the adaptable parameters of the function,
- set(params) – sets the parameters of the function to the specified values,
- getDerivative() – returns the derivative of the function (if possible),
- evaluate(input) – calculates the output of the function,

Additional learning methods are not directly included within the library, but are easily available via external scripts which can adjust any of the FCM parameters using all of the available commands. Examples of such scripts are available in repository [120].

[1] Or by inheriting and overriding the specified abstract class if the programming language does not provide a support for class interfaces.

6.2.5. Final Implementation in Python

For the final implementation, we have decided to move the development over from .NET towards the *Python* programming language, due to its wide support on the all major platforms and availability on significant representation of robot models, embedded devices and mini-computers such as Raspberry-Pi. These devices play a major role in the advancement of IoT and ubiquitous robotics, however their capabilities to run powerful albeit heavy .NET framework libraries are often limited.

Python, as a high-level, interpreted and general-purpose dynamic programming language which focuses on code readability [121], has been proposed as a viable alternative to the former .NET implementation. While Python is considered a powerful scripting language, with a possibility to execute complex commands using one-liner programs, it is also an object-oriented language with a capacity to create full-scale applications, libraries and frameworks. It is also capable of easily creating web applications and interfaces thanks to several available *Web Server Gateway Interfaces* (WSGI) such as Werkzeug, Flask or Django. This is a very welcome feature, especially in case of the creation of network interfaces for the library.

The final implementation of an FCM library, titled as a *Python Open Fuzzy Cognitive Maps Library* or *PyOpenFCM*, is based on the proposal (described in chapter 6.2.4). It is provided as a standalone Python module consisting of the following main classes [122]:

- *FCM* – main class with methods to configure concepts and to calculate FCM updates.
- *Concept* – represents a single FCM concept.
- *Config* – default configuration for FCM functions & relations.
- *IFunction* – interface for membership SISO (single input, single output) function out=f(in).
    - *PiecewiseLinear* – simple piecewise linear function.
    - *Polynome* – simple polynomial function.
    - *Sigmoid* – simple sigmoid function.
    - *Predefined* – function predefined by equation string.
- *IRelation* – interface for MISO relation (multiple input, single output) between concepts.
    - *RSimpleSigmoid* – standard linear connection with sigmoid thresholding function.
    - *R3Term* – three-term weighted connection with sigmoid thresholding function.
    - *RNeural* – implements multilayer perceptron artificial neural network.

The UML class diagram of the FCM library is displayed on Fig. 55.

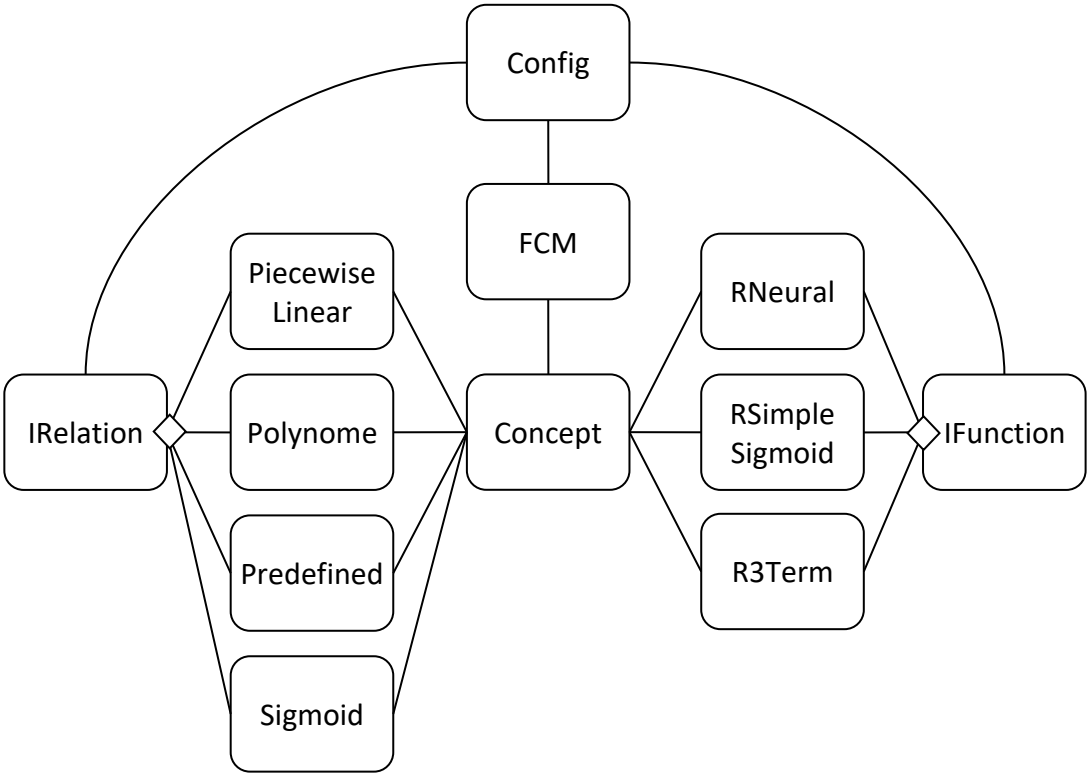


Fig. 55 Class diagram of the final implementation of PyOpenFCM library [122].

The final implementation is aimed to provide a simplicity of use by proposing several improvements. The main FCM object is represented as a dictionary of concepts that enables efficient and simple management of concepts added to the map. The respective *high-level API* of the library provides an automatic detection and creation of new concepts referenced while executing connection commands and automatically assigns default membership functions and default relations to all of the newly created concepts. Example usage of the library API is demonstrated in the following code snippet:

```
from fcmlib import FCM
map=FCM(C1=0.6,C2=0.4) # initialize a new FCM with two concepts
map["C3"]=0.5          # add a third concept activated to 0.5
map.connect("C2","C1") # connect the concept C2 with C1
map.connect("C4","C3") # create a concept C4 & connect it to C3
print(map.listPreceding("C3")) # list concepts preceding C3
```

All of the functions and relations of a concept can be, naturally, redefined subsequently using the commands from the *low-level API*. An access to individual concepts is simple, due to the implementation of FCM object as a dictionary. An example of this interaction using the default, linear weighted relation, is shown in the following code:

```
print(map["C4"].relation.get())# show relation of C4
map["C4"].relation.set("C1",0.2) # set relation of C1->C4
print(map["C4"].relation.get("C1")) # and show the change
```

The current state of activations of concepts within the map can be recalculated for each simulation step simply by calling the *update()* method. The map state can be also serialized to a JSON string or file. Similarly, a new map can be loaded from such a string or file:

```
map.update() # update FCM by propagating signals via relations
map.save("example.json") # save FCM to a file as JSON
newmap=FCM("example.json")  # load FCM from the JSON file
```

One of the disadvantages of Python is its interpreted nature, due to which Python programs may run slower than similar programs in other languages. We focused on mitigation of this effect by enabling the compilation option for the library, therefore it is compiled directly into byte code (with CPython) during the installation or at first run (if used without installation).

The speed of computations of FCM updates has been further improved by representing the map in a form of an adjacency list (see Fig. 56), instead of the traditional dense matrix representation (see chapter 3.2). An adjacency list can be considered as one of the most efficient representations to perform updates in sparse graphs such as FCM, which has been proven by our original research [123] in the sparse matrix-vector multiplication (SpMxV) benchmark [124].

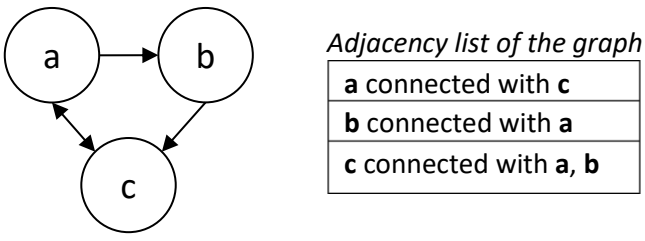


Fig. 56 Efficient representation of a graph in form of an adjacency list.

One of the main advantages of the library is its ability to utilize powerful complex relations between concepts by using three-term and neural MISO relations in addition to variable SISO membership functions for fuzzification of input concepts or defuzzification of output concepts. The setup of default relations and function for each FCM object can be done by initializing its subordinate Config object. After the default setup, all the newly connected Concept objects will use the relations and functions according to the configuration.

Another feature of the library is an integration of automatic adaptation using least mean square method and error backpropagation. A simple example of training a single neural relation is demonstrated in the following code:

```
from fcmlib.relations import RNeural
from fcmlib import Concept

r=RNeural(2) # Neural Relation (1 hidden layer with 2 neurons)
r.attach(Concept("C2",1.0))# Attaching concept C2 to relation
r.attach(Concept("C3",1.0))# Attaching concept C3 to relation
print(r.get()) # Serialized weights
print(r.propagate()) # Show result of forward propagation
r.set("0.9,1.0,1.0,1.0,1.0,1.0") # Setting new weights
r.set("C2","0.5,0.5") # Setting weights for C2 exclusively

error = 1
while error > 0.1:
    res=r.propagate()
    error=1.0-res
    r.backprop(error)
    learning_rate=1.0
    r.adapt(error,learning_rate)
```

Note that the example is being simplified to training using a single sample {1.0,1.0 → 1.0}. However, this approach can be generalized to any number of samples and various training datasets using various number of concepts within the map.

Additional adaptation methods can be implemented either as new modules (using the interfaces IRelation and IFunction) or applied by scripts utilizing the standard commands from the library API. Example of employing the semi-interactive evolutionary optimization is given in chapter 7.

6.3. Web Application Programming Interface

In addition to being implemented as a Python module, the library also provides a web application programming interface (Web API) available through a complementary service application. The application is created as a *Web Server Gateway Interface* (WSGI) using a *Flask Python Microframework* [125]. It can be used to create an *FCM server* performing remote computations for network devices including robots, sensors and actuators.

The Web API enables use of all of the library commands within HTTP requests. In addition to these basic commands, there are several additional functionalities (see Fig. 57), such as:

- creation and storage of maps on the server,
- listing of created maps,
- deletion of created maps,
- export and serialization of maps as JSON strings,
- import of maps from JSON strings.
- command line interface (CLI),
- graphical user interface (GUI).

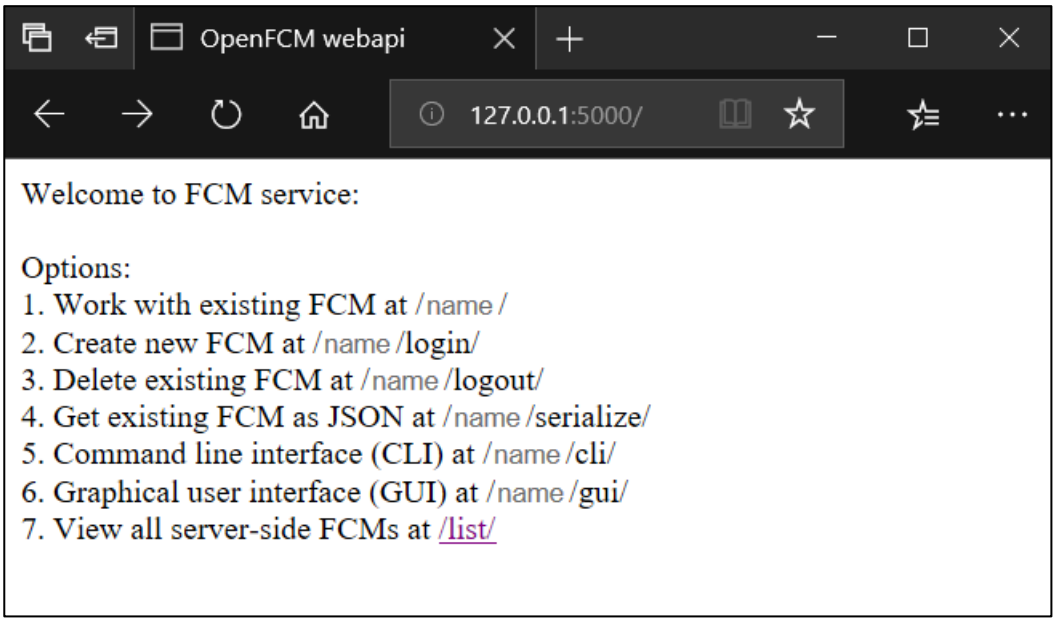


Fig. 57 Online FCM service provided by the library.

The Web API can process commands wrapped inside HTTP requests using either GET or POST methods. The command strings are then deserialized and executed on the server using Python's standard *eval* and *exec* functions, which are further safeguarded against malicious or unwanted use by restricting the Python interpreter exclusively to the methods provided by a specified FCM object

(i.e. a single map). The following code example shows a registration and work with a new map titled "*newfcm*". Executed library ***commands*** are shown in bold:

```
http://localhost:5000/newfcm/login
http://localhost:5000/newfcm/execute/newfcm["C1"]=1
http://localhost:5000/newfcm/execute/newfcm.connect("C1","C2")
http://localhost:5000/newfcm/execute/newfcm.update()
http://localhost:5000/newfcm/serialize
```

The example clearly shows, that an interaction with a single map is available through its associated URL (e.g. */newfcm/*) at locations that provide:

- ***registration*** of a new map (at */newfcm/**login***),
- ***execution*** of arbitrary library commands (at */newfcm/**execute**/command*),
- ***export*** of the map model (at */newfcm/**serialize***),
- ***deletion*** of the map (at */newfcm/**logout***).

The commands can be sent from any software program that supports creation of HTTP requests, including web browsers (see Fig. 58).

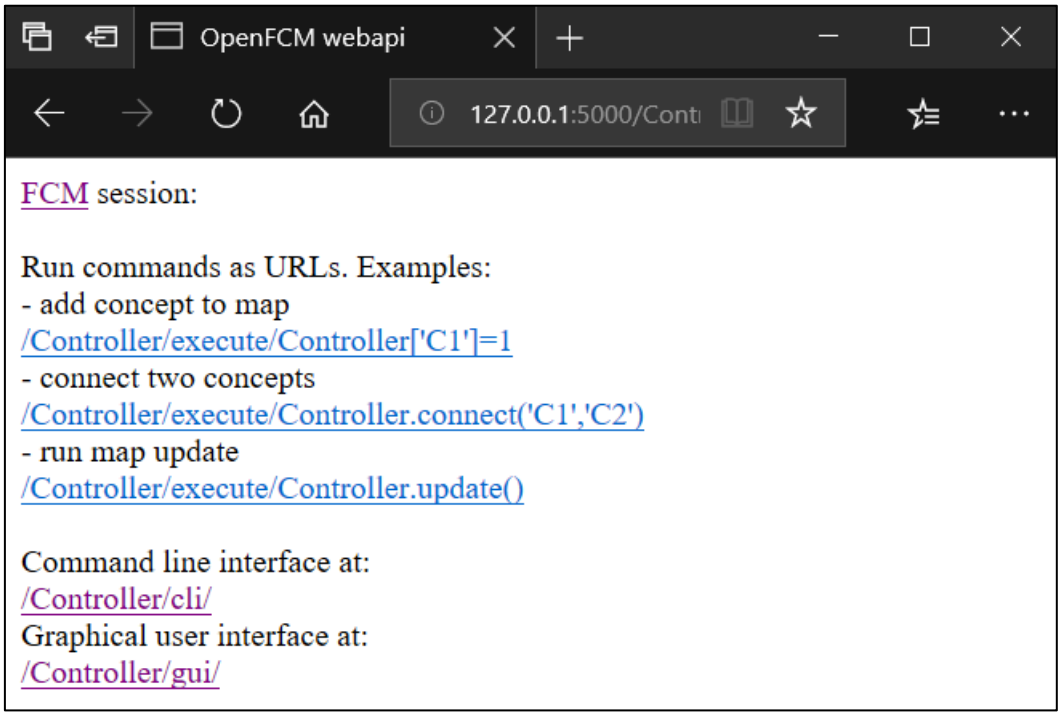


Fig. 58 Examples of web API usage in a web browser.

6.4. User Interfaces

The preliminary implementation in .NET (chapter 6.2.3) was created along with a user interface (displayed on Fig. 53). Similarly, also the final implementation is provided by several means of user interaction, including:

- interactive mode of the standard Python *interpreter*,
- web based *command line interface*,
- *graphical interface* provided by JavaScript application.

The simplest way to access library functionalities is to use the standard interactive Python interpreter, which is provided with every Python distribution. The library commands become available after a single import statement, e.g.:

```
from fcmlib import *
```

The other option to interact with the library is to use the command line interface (CLI) integrated with the Web API. The interface is available on *FCM server* at URL */mapname/cli/* and provides an online interactive Python interpreter. The interpreter is implemented as an HTML form (see Fig. 59) and can be used to send commands to the FCM server via HTTP POST method. The results of each command are displayed after its execution.

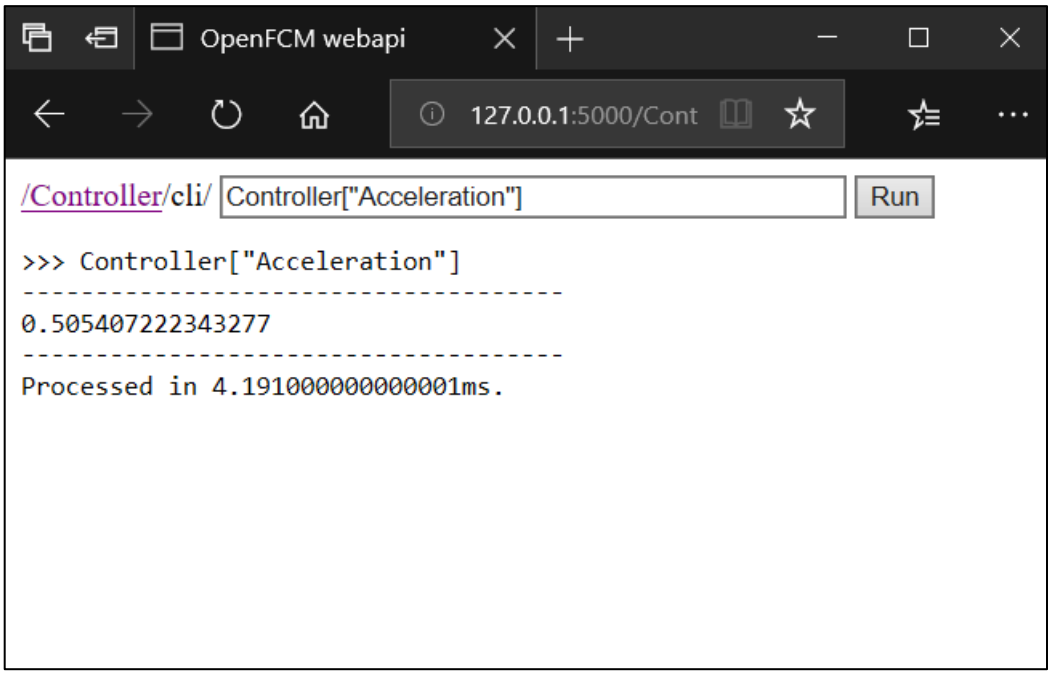


Fig. 59 Command line interface (CLI) of the online FCM service.

And finally, the user interaction is also possible through a simple GUI application (see Fig. 60), which combines CLI with a graphical visualization of the map. The visualization is provided as a JavaScript application that displays and animates an SVG image using D3 library [126].

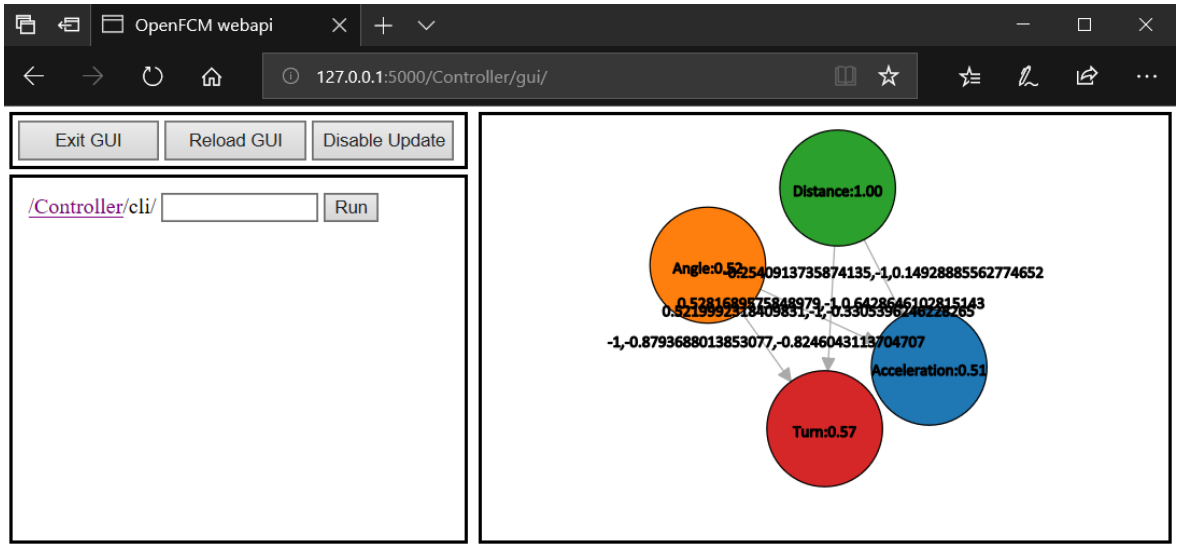


Fig. 60 Graphical user interface (GUI) of the online FCM service.

The application script is used to regularly communicate with FCM server on background and update the map graph whenever a change occurs. Since the status of the displayed map is updated in a real time, it is possible to monitor its condition even while being modified by other programs and devices, such as robots, sensors and actuators. In this way, it can be used to monitor the state of these devices.

The D3 library [125] is employed to automatically calculate the layout and distribution of the displayed concepts (similarly to the .NET MSAGL, see chapter 6.2.3). It also provides a simple point-and-drag interaction, which can be used to rearrange concepts at the discretion of the user. However, the actual user-driven modification of the map needs to be performed via commands executed from adjacent CLI. In the future, it is possible to include point-and-click support which could be used to open additional windows with possibility to directly change concept activations or redefine relations and membership functions.

7. Fuzzy Cognitive Maps for Control of Intelligent Space

"Invention is the most important product of man's creative brain. The ultimate purpose is the complete mastery of mind over the material world, the harnessing of human nature to human needs."

Nikola Tesla

All of the ideas presented in previous parts of this work naturally lead to a synthesis claiming several goals. The ***third contribution*** of this dissertation aims "t*o propose a situational control model of an intelligent space utilizing FCMs."* While employing this general model, we continue with the ***fourth contribution*** which objective is "*to propose a control structure of an intelligent space for a situational class of robot navigation."* And finally, we conclude our effort with the ***final contribution*** that aims *"to create a simulation of a situational control of elements of an intelligent space."*

7.1. Situational Control of Intelligent Space

We will begin this chapter with an overall proposal for situational control of intelligent space, hence satisfying the ***third contribution*** (*"the proposal for a situational control model of an intelligent space utilizing FCMs").*

Since a *situational control* (as introduced in the chapter 1.4) is a complex methodology that expects extensive preparations and planning in advance to the consecutive system operation, it requires to consider a wide range of hypotheses about a behavior of the system, its control processes and its global objectives. The result of this preliminary *planning phase* is the set of applicable control strategies for the next *operational phase.* In the following lines, we will try to gradually materialize all of the necessary steps relevant in order to a satisfy this methodology.

7.1.1. Description and Global Objectives of Intelligent Space

Before an attempt to propose a control framework for intelligent space, it is necessary to follow the basic steps according to the methodology of situational control (according to the chapter 1.4.1), which include:

1) a *description of the controlled system*,
2) the *setting of global control objectives*,
3) and a *proposal of situational classes* (frames).

With regard to the *first point*, this proposal will consider the Intelligent Space [71] at the Center for Intelligent Technologies (CIT) [87] which was *already described* in chapter 5.2.2. As it was previously noted, the space includes several types of sensors including cameras and microphones, ambient sensors for various physical values, and finally, several types of mobile robots. In order to simplify the overall proposition, we will not deal with networking layer for this moment and expect that it is optimized and working according to the needs of all the attached devices.

The *goal objectives* and requirements for control of IS can be characterized in several levels:

- First of all, the control should satisfy the needs of its inhabitants being either humans or robots, while humans should always have a priority in case of any conflict.
- Secondly, the proposal should comply with current legislation and internal organizational rules, if used to automatically generate necessary documentation (including attendance books, visitor lists, cleanup records, etc.).
- And finally, the system should aim to minimize energy and resource consumptions by employing optimized control strategies and algorithms.

7.1.2. Proposal for Situational Decomposition of Intelligent Space

There is a large number of conditions or *situations* which can determine the current state of the intelligent space. The number of situations is theoretically unlimited, however, in practice it depends on applied use cases (see chapter 5.2.3 and 5.2.4). In order to apply the methodology of situational control, these situations need to be categorized into classes, where each class consists of several related and similar situations. The proposal for this situational decomposition is displayed in Fig. 61.

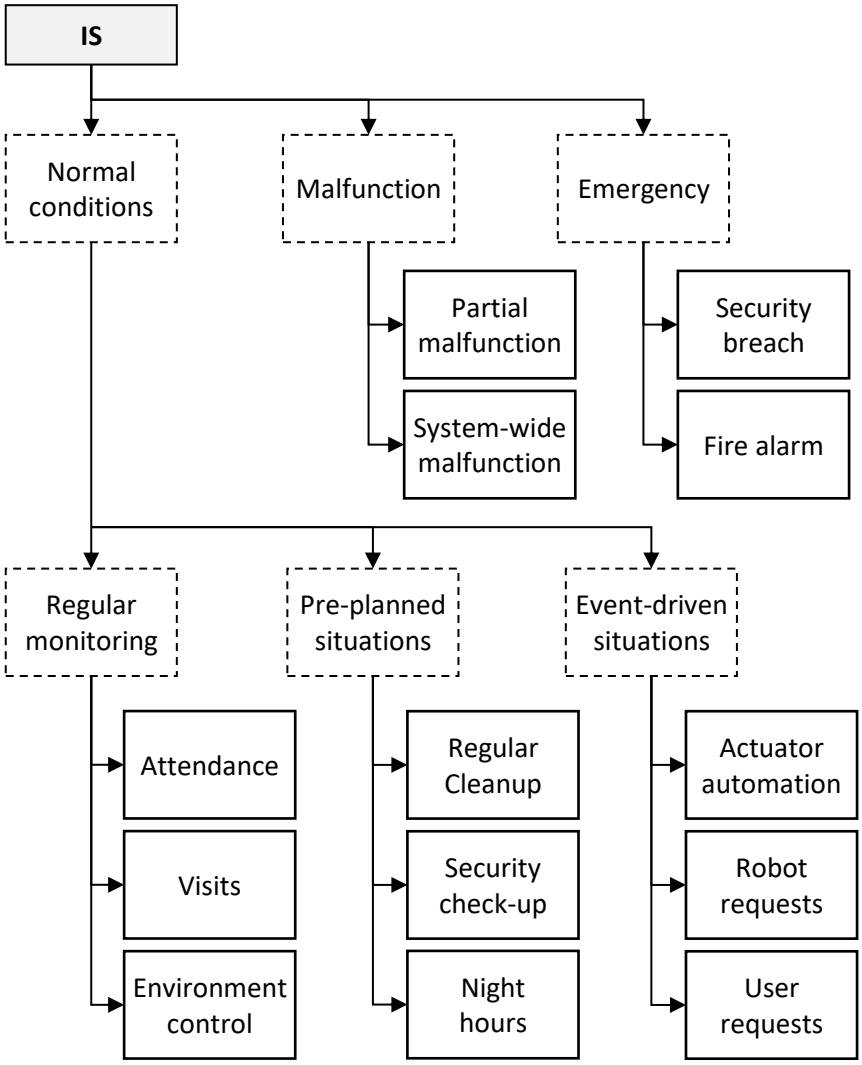


Fig. 61 Proposal for situational decomposition of IS.

According to the decomposition, the IS can find itself in several conditions. For *normal working conditions*, we propose decomposition into three situational frames, i.e. a *regular monitoring* of the IS, an execution of *pre-planned situations* and reactions to various *events* happening within the IS. It should be also noted, that it is possible to experience several concurrent situations in the same time (e.g. visits, cleanup, temperature control, etc.).

During a *regular monitoring*, the IS proceeds to record common activities of its inhabitants, such as *attendance* monitoring by logging arrivals, departures and breaks of employees. Additionally, the system can record activities of other *visitors*, i.e. students, passers-by and other persons. These situations do not necessarily require any feedback from the system (using actuator devices), but merely produce logs which are stored in central database for later use. On the other hand, long-term continuous environment monitoring may result in some control actions, such as heating or air conditioning adjustment.

Pre-planned situations cover all the activities, that happen in regular intervals. These may include regular *cleanups* of predefined areas using janitor robots, or *security patrols* by safety drones. Moreover, pre-planned conditions may include significant change in system operation, such as complete area lock-down and energy saving mode during *night hours*.

Event-driven situations comprise of short-term reactions of the IS to various inner or outside stimuli. An example of such event may be a *user request* to utilize specific resource or service provided by the IS. This can be a request of a visitor for guidance towards a specific destination which can be either office or a certain employee located within the IS. The IS may react to user behavior even without any imperative, by *automating* certain *actuators*. A typical example is automatic unlocking or even opening doors in proximity of privileged users. In addition to users, the IS may provide additional services according to *robot requests*, including precise localization, optimal path planning and navigation (see chapter 7.2).

Obviously, in addition to normal working conditions, the IS can find itself in a state that severely deviates from these conditions. The deviation may be caused either by an internal *malfunction* of the system or an external threat (*emergency*). System errors can be either *partial*, in that case only a few selected elements (sensors, actuators or robots) do not work properly, or *system-wide*, where the function of the system as a whole is threatened (e. g. in case of a network failure or a database corruption). Emergencies can be caused either by the environment (such as *fire alarm*) or by deliberate attack on the system (i.e. *security breach*).

7.1.3. General Strategy for Control of Intelligent Space

After the specification of individual situational classes, it is necessary to propose the algorithmic solutions that are suitable to manage these situations with regard to the global objectives of the system. The methodology of situational control therefore proceeds with these further steps [1]:

4) Description of control strategies assigned to each situational class.
5) Algorithmization of individual control strategies.
6) Implementation.

In order to control the IS, we propose to utilize the general model of the *formatter control* (according to the chapter 2.3, Fig. 18), which is suitable to be applied with a range of distributed IoT sensors, that can be linked to associated *analyzers* providing information for the central *database* (see Fig. 62).

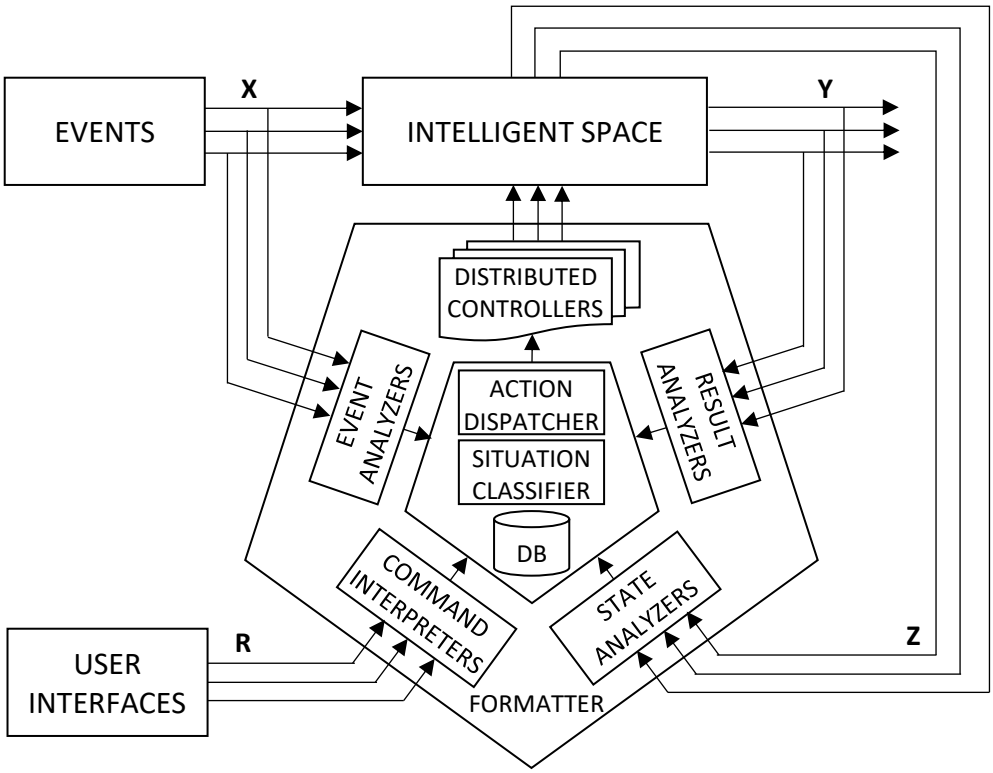


Fig. 62 Proposed model for control of IS.

For further control, we propose a modern *event-driven* approach [127] which is currently a default paradigm in software engineering. *Event logs* stored in the database are evaluated by appropriate *event handlers* with regard to a specific situational class. Further actions are then initiated by an *action dispatcher* and executed using a range of distributed *software controllers* (implemented as network services either within fog or edge computing layers) which control actual devices within the IS, including actuators and robots.

Algorithmization of individual components of the formatter (i.e. analyzers, database, dispatcher and controllers) can be done using several methods, which implementation depends on a specific component:

- *Analyzers* gather the raw information from the IS in a form of video, audio or numeric data and convert it to appropriate events. For example, cameras may provide constant video streams, which can be analyzed in order to detect movement, identify objects and persons or provide localization services. Ambient sensors, such as thermometer provide simpler data in a form of numerical series, however, even such information can be analyzed in order to detect deviations from a desired state. Implementation of various analyzers can be done either using simple *rule-based methods*, *fuzzy systems* including *FCMs*, *neural networks*, or *specialized algorithms* such as Tracking-Learning-Detection [128].

- *Database* is necessary in order to secure proper information processing for the IS. It is used to gather all the data and event logs provided by the IS. It can be implemented using several database systems, including *SQL* and *NoSQL* alternatives.
- *Situation classifier* determines the current situation class according to the information from one or more analyzers. In case of parallel situations, it can consist of several situation detectors. Its implementation can be done using either rule-based classifiers, *neural networks* or even *FCMs*.
- *Action dispatcher* module is responsible for an initiation of respective control strategies according to the current situation. In most cases it can be implemented as *rule-based* decision-making system, however it is also possible to utilize *fuzzy cognitive maps* with the same result.
- *Controllers* are accountable for an execution of individual control strategies. These can be implemented either as traditional industrial *PID controllers*, *fuzzy controllers*, *neuro-controllers*, *FCMs* or other appropriate alternatives. It is important to notice, that these controllers can also have a hierarchical structure, which means that they themselves can be composed of several co-operating elements. The controllers can be responsible for either individual actuators or mobile robots (an example of a controller for navigation of a mobile robot is given in the next chapter).

In addition to the basic operational components of the formatter, there are also several additional algorithms, which are necessary in order to ensure a complete functionality of the system. These additional components include *user interfaces*, which provide a possibility to send direct commands to the system, along with long term *analytics services* (or data-mining services), which may try to optimize any of the individual processes responsible for achieving the global goals of the system. And finally, the individual control strategies need to be parametrized according to the specific situation, which is provided by various *optimization and adaptation algorithms* based on the methods of either supervised or unsupervised learning, including evolutionary optimization.

After the initial *planning phase* and implementation of situational control, the system is put into operation and further managed in an *operational control phase*, that uses three decision-making nodes [1]:

7) The classification to the appropriate class of situations (using *situation classifier*),
8) The choice of a corresponding control strategy (using *action dispatcher*),
9) The realization of particular control functions (using one or more selected *controllers*).

It should be noted, that an *implementation of all of the* ***necessary algorithms*** *is way* ***beyond the possibilities*** *and capacities of a* ***single person***. We can clearly estimate, that in order to implement all of the functionalities required to provide situational control for the IS (according to the proposal in chapter 7.1.2 and Fig. 61), it would be necessary to define algorithms for *hundreds* of individual control elements and even more in order to ensure their proper mutual communication over the network. This is a task for a dedicated team consisting of *dozens of developers*, or for long term student projects happening over a course of several years. Therefore, with regard to this work, we have proposed and realized only a *particular set of selected control functions* necessary for a ***navigation of a mobile robot*** within a limited range of situational classes.

7.2. Navigation of Mobile Robot in Intelligent Space

In this chapter we describe a possible realization of particular control functions necessary for *navigation of a mobile robot*. This *control strategy* is required in several situational frames (proposed in chapter 7.1.2, Fig. 61) including robot performed *cleanups*, *patrols* and *user requested assistance*.

A *navigation* can be defined as a process of monitoring and controlling the movement of a mobile robot (or any other mobile object) from one place to another. It can be executed using these steps:

- *localization*,
- *path planning*,
- *movement planning and control*.

The first step towards a successful navigation is a precise *localization* of both a current position of a robot and its destination. The localization of a robot can be performed using several options. Internal localization sensors such as gyroscopes, accelerometers or odometry can be used to calculate incremental changes in position with no need to any external assistance. However, these methods suffer from gradual error accumulation and do not allow for long term operation. Ubiquitous robots therefore often utilize various forms of external localization systems, including RFID tags, Bluetooth E-Beacons, Wi-Fi signal maps or even GPS when used outdoors [90]. Similarly, object detection and tracking using *cameras* can be used for the same purpose as well.

Specialized algorithms for a *path planning* are also often necessary, since the environment, in which a robot does need to maneuver, is usually very diverse with a lot of potential obstacles. While there are several approaches to specify proper movement trajectory, a *waypoint-based path planning* is probably one of the simplest solutions and it is used in a majority of applications. An example of this approach is displayed on Fig. 63.

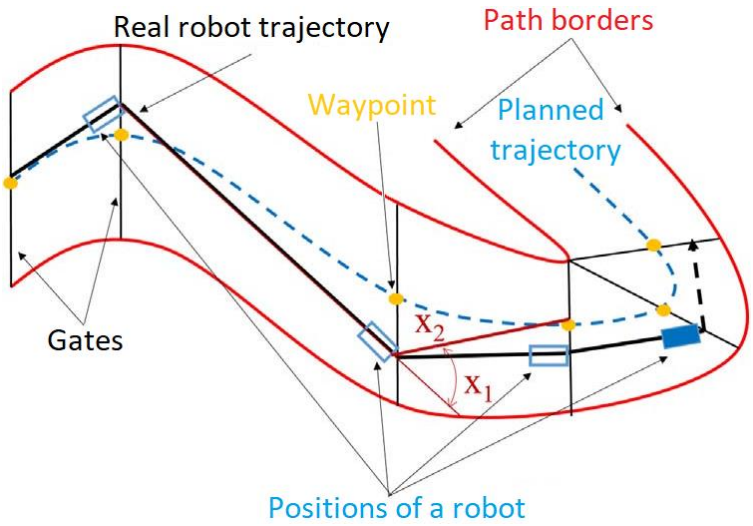


Fig. 63 Sampling of a path into waypoints with indication of inputs for robot controller: x_1 is the angular deviation and x_2 is the distance magnitude [59].

In order to *control* a robot moving on a pre-planned, waypoint-based trajectory, two separate control elements are necessary:

- A *navigator* component is used for: a proper selection of a current waypoint; switching between waypoints based on their distance; and a calculation of an angular deviation from the optimal trajectory.
- A *controller* component is used to directly actuate various motors and other actuators of the robot in order to keep it on a track towards its destination. The control actions are based on the information from the navigator component (such as an angular deviation and a distance from the current waypoint, current acceleration, actual speed, etc.)

7.3. Fuzzy Cognitive Maps in Robot Control

Based on the assumptions from the previous chapter, we have proposed an *FCM controller* of a *wheeled* mobile robot (such as Qbo or TurtleBot). The proposal assumes that the path, in a form of *waypoints*, is assigned to the robot beforehand, by a superior *path-planner* algorithm. It also expects utilization of a *navigator* component, that can lead the robot between waypoints and produce input values for the controller including an *angular deviation* and a *distance* from the next waypoint. The controller is hence able to modify robot's *acceleration* or its *deceleration* along with its *turning rate*, see Fig. 64.

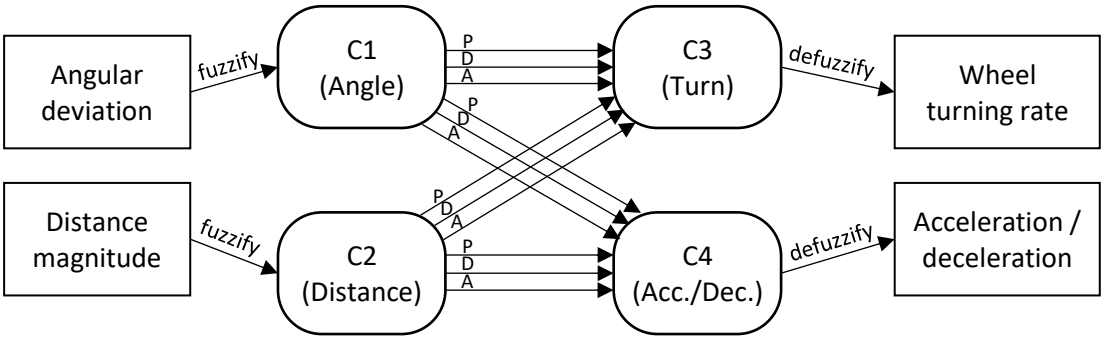


Fig. 64 Three-term FCM controller of a wheeled mobile robot.

Membership functions for fuzzification of inputs and defuzzification of outputs need to be set by an expert. In our experiments, the functions were defined as follows:

$$f_{C1}^{in}(x) = \begin{cases} 0, & -90 > x \\ 0{,}5 + x/180, & -90 \leq x \leq 90 \\ 1, & x > 90 \end{cases}$$

$$f_{C2}^{in}(x) = \begin{cases} 0, & 0 > x \\ x/40, & 0 \leq x \leq 40 \\ 1, & x > 40 \end{cases}$$

$$f_{C3}^{out}(x) = 90x - 45$$

$$f_{C4}^{out}(x) = 20(x - 0{,}5) \qquad (17)$$

The controller utilizes *three-term relations* in order to reduce a number of necessary concepts to a minimum. Optimal values of relational weights are to be found by a semi-interactive evolutionary algorithm (using an approach presented in the chapter 4.3.2) applied within a simulated version of the intelligent space (see a chapter 7.5). The simulation is based on a paradigm of situational modeling (according to the chapter 1.4.2).

Localization of the robot is expected to be determined externally, using camera detection. The *location analyzer* converts a set of 2D coordinates of a robot, detected within an image, into a respective set of 2D coordinates related to the plan of the intelligent space. The conversion is performed by an FCM with neural relations according to Fig. 65.

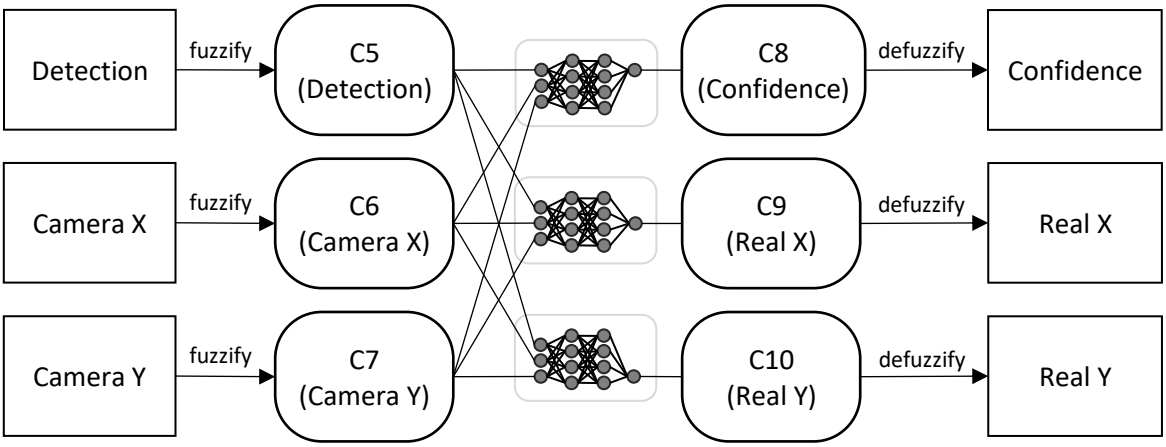


Fig. 65 Location analyzer based on FCM with neural relations.

If we assume the resolution of the camera image as $r_x \times r_y$ and the size of the space as $s_x \times s_y$, then the *membership functions* can be defined simply as linear functions:

$$f_{C5}^{in}(x) = x \qquad f_{C5}^{out}(x) = x$$

$$f_{C6}^{in}(x) = x/r_x \qquad f_{C6}^{out}(x) = xr_x$$

$$f_{C7}^{in}(x) = x/r_y \qquad f_{C7}^{out}(x) = xr_y$$

$$f_{C8}^{in}(x) = x \qquad f_{C8}^{out}(x) = x$$

$$f_{C9}^{in}(x) = x/s_x \qquad f_{C9}^{out}(x) = xs_x$$

$$f_{C10}^{in}(x) = x/s_y \qquad f_{C10}^{out}(x) = xs_y \tag{18}$$

The neural relations are to be trained by a supervised learning using a pre-recorded dataset of robot detections within a camera image and respective locations within the IS. Such dataset should be created beforehand, during a *planning phase* of *situational control*, for every camera employed within the space.

7.4. Proposal for Navigation System in Intelligent Space

In this chapter we present the ***fourth contribution*** of this dissertation, which aims *"to propose a control structure of an intelligent space for a situational class of robot navigation."* There are several devices and software programs necessary to realize a complete and comprehensive *control strategy* for the purpose of *navigation of a mobile robot*. The devices include:

- *mobile robot* which operates within the IS,
- one or more *cameras* that overview the entire operational area of the robot,
- and selected *actuators*, e.g. self-opening doors, necessary to allow unrestricted movement.

Required software programs include:

- *detection analyzers* that acquire robot coordinates within camera images,
- *location analyzers* that provide a robot localization within the IS,
- a *navigator service* that guides a robot on a pre-defined path,
- and a *controller service* that generates control signals for robot motors.

In addition, there are several superior services on a higher level of control hierarchy, including *path planner*, *action dispatcher* and other *formatter modules* (according to Fig. 62). A generalized scheme of basic components realizing the *control strategy* is displayed on Fig. 66. Services that can be implemented as FCMs are shown in gray boxes.

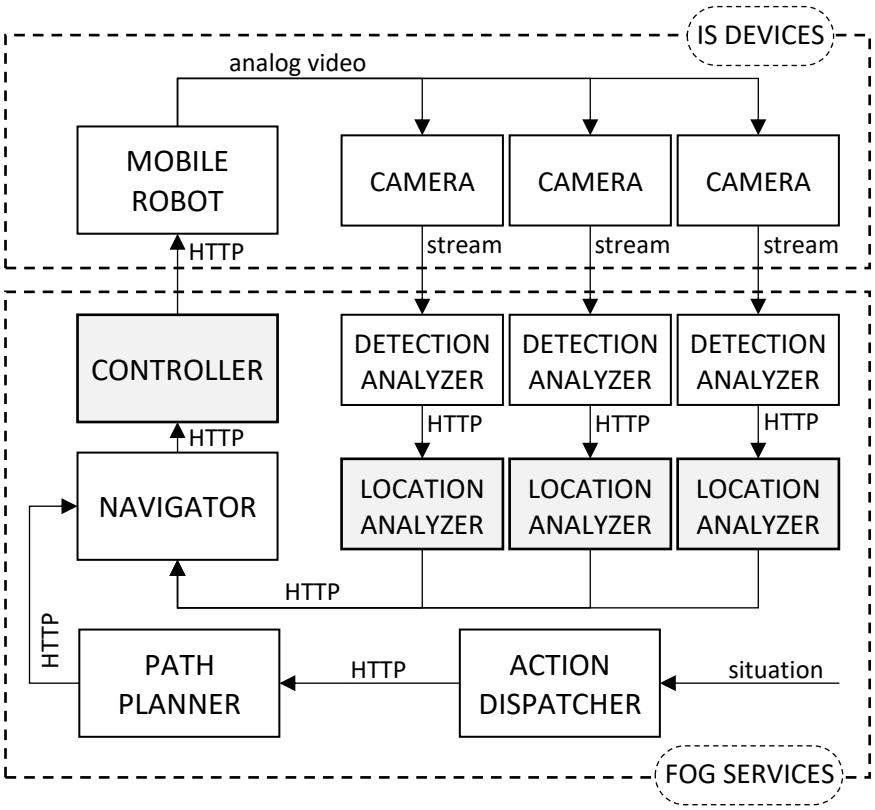


Fig. 66 Components of a navigation system within IS. FCM services are displayed in gray boxes.

The initiation and execution of the control strategy for robot navigation proceeds as follows:

1) A specific situation (such as *user request*) evokes the *action dispatcher* to initiate the control strategy for robot navigation.
2) After a consideration of an initial position of the robot and its destination, a *path planner* module generates a waypoint-based route, which is dispatched to a *navigator* component.

3) A *navigator* component subscribes to all of the necessary services, including *controller* and *localization* FCM services.
4) A *communication* between individual devices and services is realized directly using a *subscription-based paradigm* (via direct HTTP requests). Since all of the required modules establish a direct mutual communication, extensive database lookups are avoided, which mitigates a risk of possible communication bottleneck through a database.
5) The control strategy proceeds until the robot reaches its destination.
6) Finally, a *navigator* component generates a *termination event*, unsubscribe from the linked services and conclude the control strategy. The IS continues to operate in normal working conditions.

7.5. Experiments in Intelligent Space Model

In order to further examine the proposed control strategy and evaluate a feasibility of application of all of the implemented FCM-based services, we have created a simplified model [120] of the intelligent space (proposed in the chapter 5.2.2). The created simulation accounts for the ***final contribution*** of this dissertation, that aims "*to create a simulation of a situational control of elements of an intelligent space.*"

The simulation implements a dedicated *control strategy* based on *fuzzy cognitive maps* for a mobile robot in a *situational frame* of *robot navigation* within an intelligent space. It is centered around a simulated robot implemented as a wheeled vehicle with control elements being the current acceleration (or deceleration) and a current turning angle (according to the chapter 7.3). In order to provide a localization of the robot, there are eight standalone *cameras* situated in various spots within the IS. Each camera is provided with its respective *location analyzer* service based on FCM model. The robot control is performed by two main components: a *waypoint-based navigator* and an *FCM-based controller*. The robot follows a pre-defined track and is controlled using these main steps:

1) Each of the FCM-based *location analyzers* provides an estimated location of the robot.
2) The *navigator* selects the most accurate location estimation (considering the confidence of the analyzer) and approximates the current trajectory (using the last two known locations of the robot).
3) The *navigator* calculates a current distance from a next waypoint along with an angle deviation of the current trajectory.
4) The FCM-based *controller* uses the distance and angle deviation as inputs and subsequently applies a correction to the control elements of the robot.

A visualization of the simulated model is shown on Fig. 67. It shows a wheeled mobile robot driving through a pre-defined path while employing localization services provided by stationary cameras according to the paradigm of ubiquitous robotics.

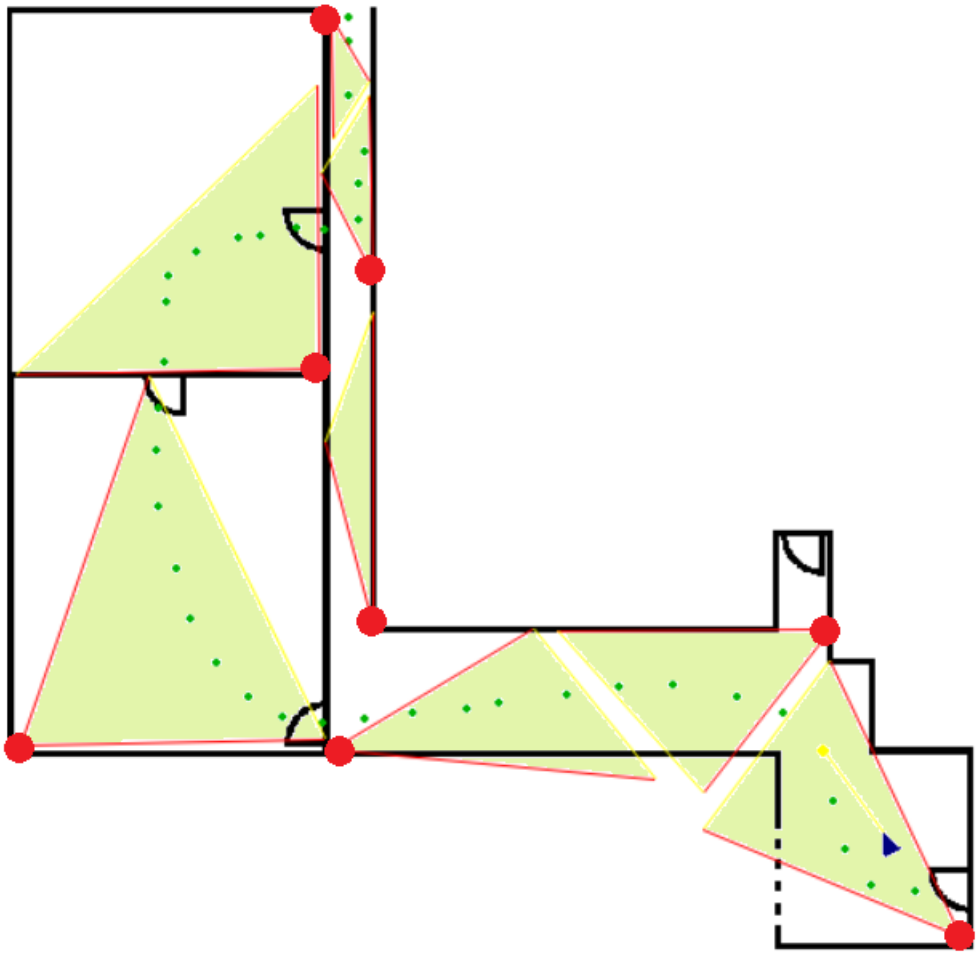


Fig. 67 A simulated model of IS. Cameras are shown in red, their respective vision fields are shown in light green. A robot is displayed as a blue triangle. Its path is defined by waypoints marked as dark green dots [120].

Before the actual use of the control strategy, the simulation requires a manual preparation of these pre-requisites:

- definition of robot *path* and placement of *cameras* (using a script "*setupmap.py*"),
- training of *FCM* based *localizers* (using a script "*traincams.py*"),
- training of *FCM* robot *controller* (using a script "*traincontroller.py*").

The first step actually deals with a basic environment setup. The simulator [120] provides this setup in a form of a script "*setupmap.py*", which shows a plan of the IS and enables user to define both the proposed robot trajectory along with locations and vision fields of cameras.

The second step is to train FCM models of *location analyzers*, which can be carried out by a script *"traincams.py*". The location analyzers provide 2D coordinates of the robot within the IS, provided by a transformation of robot detections within camera images. The analyzers are based on FCM models with neural relations and are trained using LMS error backpropagation that is natively supported by the PyOpenFCM library. The script generates the training samples in a simplified way, considering the nature of the simulation, as a randomized pair of input and output 2D coordinates.

As the lasts step of the preparation phase, an FCM controller of a mobile robot is created using a *semi-interactive evolutionary optimization* provided by a script *"traincontroller.py*". A screenshot of an evaluation interface is displayed on Fig. 68. The optimization is performed using the same approach that was already described in the chapter 4.3.2.

In order to accelerate the preparation phase, all of the training scripts utilize an offline API version of the PyOpenFCM library and are executed as standalone programs. The results of these scripts, in the form of FCM models, are saved as JSON files and can be loaded and used anytime during an operational control of the IS. During this phase, all of the created models are provided online as network services. An access to these services is available via an FCM server implemented using Web API of the PyOpenFCM library proposed in the chapter 6.2. An example of a single simulation run is displayed on Fig. 69. A complete video recording of this simulation run is available at [129].

As a side note, it should be admitted, that the simulation model is slightly simplified by assuming the automation of additionally required actuators, such as self-opening doors and by ignoring potential obstacles that may occupy the space. In future applications, these problems are supposed to be solved by other control strategies employed within the general scope of the situational control of the IS.

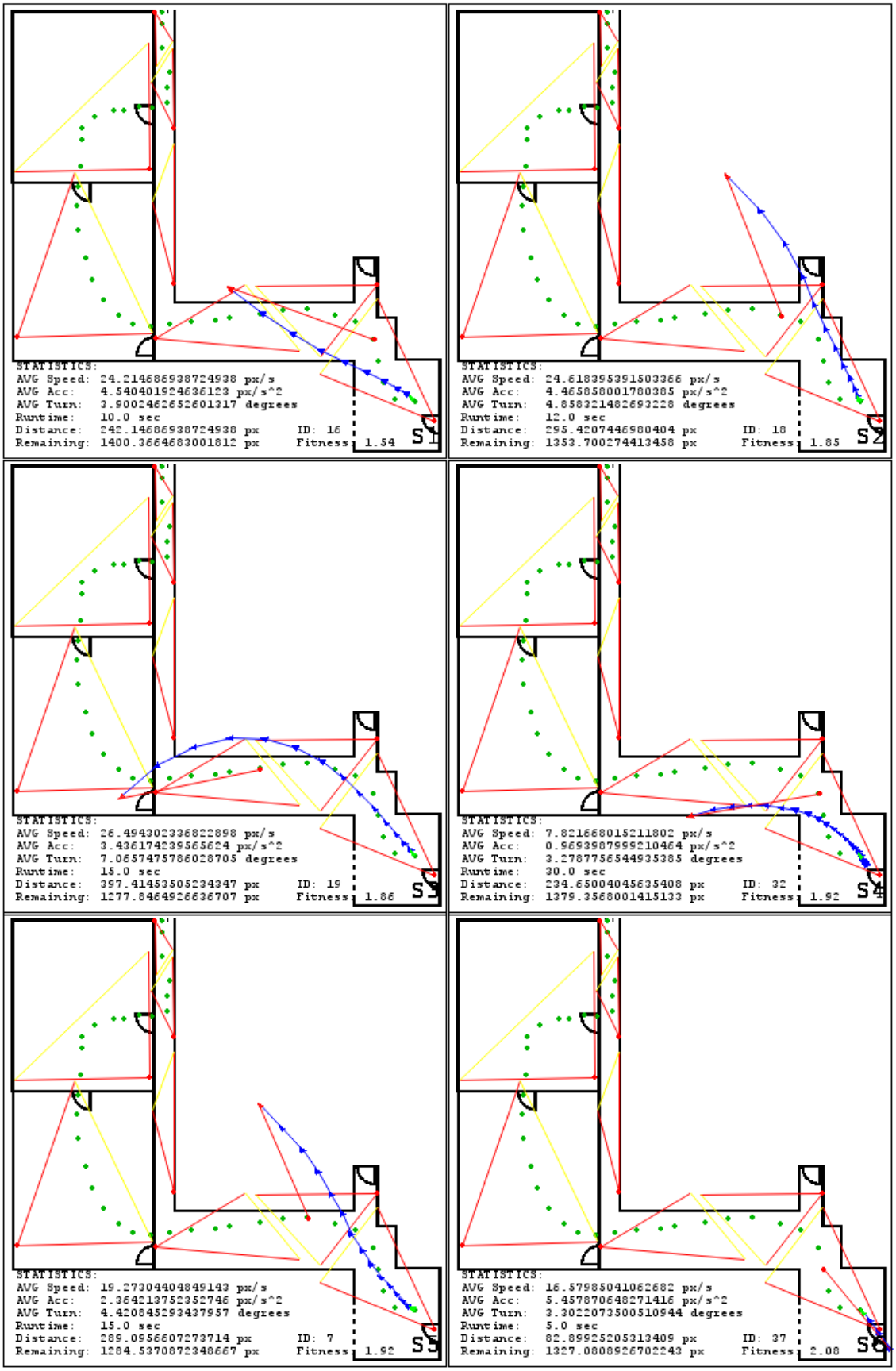


Fig. 68 An evaluation GUI for semi-interactive evolutionary optimization of FCM-based controller of a mobile robot navigated through the intelligent space [120].

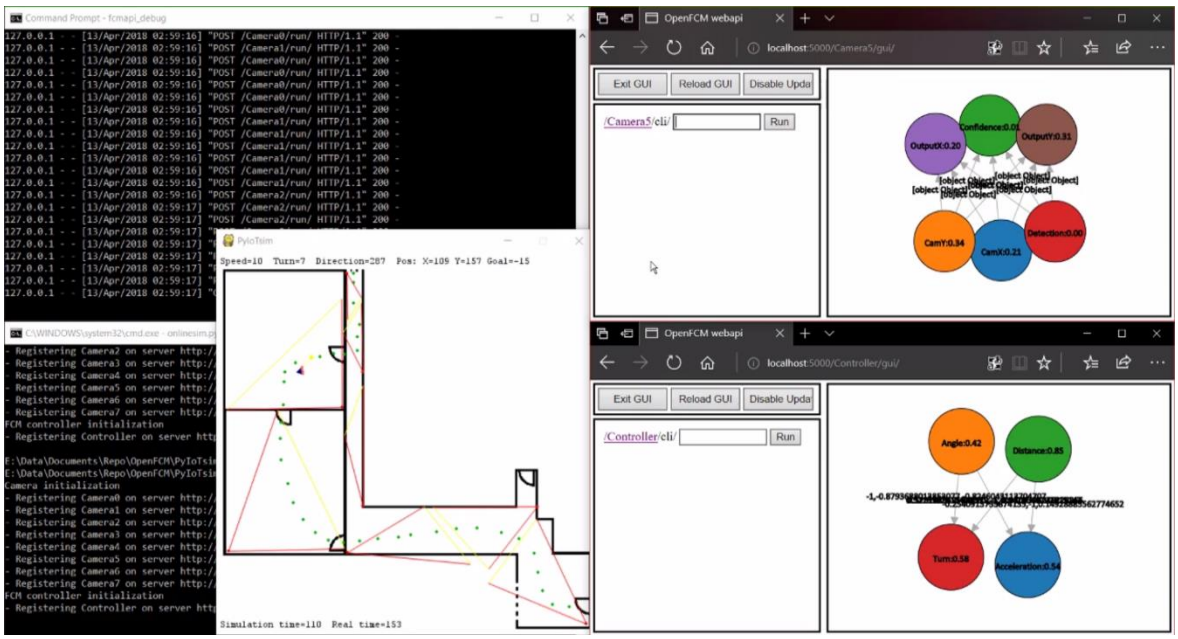


Fig. 69 An overview of simulation of a situational control strategy for navigation of a mobile robot through the intelligent space [120]. The overview shows a selection of software programs participating on the control strategy. A bottom-left window displays a simulator of the IS, along with a graphical representation of the space. A top-left window displays communication with associated FCM server. A top-right window shows a location analyzer for a single camera and a bottom-right window displays a robot controller.

Our experiments with a simulation have concluded with several findings. While the experiments have proven the overall *applicability of the proposed control strategy* for *robot navigation*, it is important to admit, that an overall accuracy of robot movement could be improved. However, this was in line with our expectations, since the simulation was very demanding on computing resources. We have executed a whole simulation on a single computer processing several scripts running in parallel. One of those was the simulated environment, second was the FCM controller and additionally, there were eight simulated cameras with complementary location analyzers. This resulted in an average simulation step of 1 second. This time can be theoretically improved to the order of milliseconds, hence improving the overall precision of the controller. Since all of the employed FCM based components are implemented as network services using the Web API of the PyOpenFCM library, every model can be potentially implemented on a separate server, within either edge, fog or cloud computing layer. This can effectively increase the overall performance of the solution.

With regard to these findings, we propose further improvements including a refinement of FCM models used for both the location analyzers and a robot controller, and a distribution of computation within the IS. The implementation should be also verified in real-world conditions.

Conclusion

"If we wait for the moment when everything, absolutely everything is ready, we shall never begin."

Ivan Sergeyevich Turgenev

In the scope of this work we have dealt with a wide range of scientific and engineering areas. We initiated the discussion with an overview of *control and modeling of complex systems* and continued with considerations of potential utilization of *computational intelligence* in this area.

We specifically focused towards applications of *fuzzy cognitive maps* (FCM) and proposed various *extensions* to its basic design (including *three-term* and *neural relations*) with a goal to improve its applicability to model and control complex systems. In addition to expert oriented FCM design, we introduced additional possibilities for *automatic adaptation* of FCMs using either *LMS methods* from the area of neural networks or *evolutionary optimization*.

In the following parts of the works, we provided a brief introduction to the *internet of things* (IoT), *intelligent spaces* and *ubiquitous robotics*. With regard to these modern paradigms, we have demonstrated unambiguous confirmation of the *intelligent space* as a *complex system*, hence arguing the possibility to use all of the methods, applicable to control these systems, mentioned in the previous chapters.

We further discussed background technologies that make implementation of these paradigms possible, including basics of modern network *communication* and its associated computing models such as *cloud*, *fog* and *edge* computing. A range of possible *applications* for both IoT and intelligent spaces was mentioned along with a specific *proposal for intelligent space* with its detailed description.

With regard to our findings, we composed the *requirements for software* tools necessary to implement control structures in scope of IoT and intelligent spaces. We have proposed and implemented a *novel FCM library* and ensured its network application by providing a corresponding *web API*.

Finally, we analyzed possible use cases off intelligent spaces and provided a proposal for its *situational decomposition* along with a framework for *situational control* using a *formatter* composed of various network services. In order to demonstrate applicability of the proposed control structure, we have implemented a simulation with an exemplary model of *control strategy* for the situational frame of *robot navigation*.

The *main contributions* of this dissertation can be listed as follows:

- The *successful simulation* of intelligent space has confirmed the *applicability of FCMs* to model and control complex systems.
- We have accomplished a significant *simplification of FCM design* with help of a *novel FCM library* supporting designs with a use of complex *three-term* and *neural relations* between individual concepts.
- In addition, we have demonstrated feasibility of *automatic adaptation of FCMs* using either *LMS error backpropagation* or *evolutionary optimization*.
- And finally, we have proven applicability of FCMs as *remote controllers* available via network either in fog or cloud computing layer.

The future work will consist of exploring the options to further *improve FCM adaptation* focused on optimization of speed of training and improving accuracy of created models. It is also worth to focus on *improving the efficiency of network controllers*, by optimizing the hierarchical structure of network devices and services participating on formatter control, thus reducing latency and amount of data transferred across the network. Naturally, the ultimate goal should be to *implement all of the proposed algorithms in real conditions*, and possibly commercializing the ideas.

References

[1] Madarász, L., Vaščák, J., Andoga, R., Karoľ, T. 2010. “Rozhodovanie, zložitosť a neurčitosť,“ pp. 396, Elfa s.r.o. Košice. ISBN 978-80-8086-142-1.

[2] Madarász, L. et al. 2012. “Systémová analýza a syntéza,“ Elfa, s.r.o, pp. 303, ISBN 978-80-8086-193-3.

[3] Sarnovský J., Madarász, L., Bízik, J., Csontó, J. 1992. “Riadenie zložitých systémov,“ pp. 384, Alfa, Bratislava. ISBN 80-05-00945-3.

[4] Madarász, L. 2005. “Inteligentné technológie a ich aplikácie v zložitých systémoch,“ pp. 346, Elfa, s.r.o., Košice, FEI TU. ISBN 80-89066-75-5

[5] Andoga R. 2006. “Hybridné metódy situačného riadenia zložitých systémov,“ Doktorandská dizertačná práca, pp. 120, KKUI FEI TU v Košiciach.

[6] Madarász, L. 1996. “Metodika situačného riadenia a jej aplikácie,“ pp. 212, Elfa Košice. ISBN 80-88786-66-5

[7] Madarász, L. 1982. “Základné princípy situačného riadenia a formalizácie rozhodovacích procesov pri riadení zložitých hierarchických systémov.“ Kandidátska dizertačná práca, pp. 95, EF VŠT Košice.

[8] Madarász, L., Főző, L., Andoga, R., Bučko, M. 2010. “Základy automatického riadenia. Lineárne dynamické systémy - teória a príklady,“ Elfa, s.r.o., Košice, pp. 402, ISBN 978–80-8086-162-9.

[9] Jadlovská A., Jadlovská S. 2013. “Moderné metódy modelovania a riadenia nelineárnych systémov,“ FEI TUKE, ISBN 987-808086-228-2.

[10] Várkonyi-Kóczy, A., Kovácsházy, T. 1998. “Anytime Algorithms in Embedded Signal Processing Systems.“ Ix. Europen Signal Processing Conf., EUSIPCO-98, Rhodos, Greece, Sept. 8.-11, Vol. I, pp. 169-172.

[11] Madarász, L. 1985. “Riadenie organizačných systémov“, Alfa, Bratislava, pp. 267.

[12] Olej, V. 2004. “Modelovanie ekonomických precesov na báze výpočtovej inteligencie.“ M&V, Milady Horákovej 262, Hradec Králové, pp. 160, ISBN 80-903024-9-1.

[13] Sinčák, P., Kostelník, P., Novotný, P. 2001. “Strojová inteligencia, inteligentné technológie – súčasnosť a budúcnosť.“ Riadenie a informatika, FEI TU Košice. ÚTRaR SAV Bratislava, pp. 1-5.

[14] Sinčák, P., Andrejková, G. 1996. “Neurónové siete – Inžiniersky prístup. Dopredné neurónové siete.“ I. diel. Elfa, s.r.o., Košice, pp. 110, ISBN 80-88786-38-X.

[15] Sinčák, P., Andrejková, G. 1996. “Neurónové siete – Inžiniersky prístup. Rekurentné a modulárne siete.“ II. diel. Elfa, s.r.o., Košice, pp. 63, ISBN 80-88786-42-8.

[16] Cybenko, G. 1989. “Approximation by Superpositions of a Sigmoidal Function,“ Math. Control Signals Systems, 2, pp. 303-314.

[17] Hornik, K., Stinchcombe, M., White, H. 1989. “Multilayer Feedforward Networks are Universal Approximators,“ Neural Networks, 2(5), pp. 359-366.

[18] Funahashi, K. 1989. “On the Approximate Realization of Continuous Mapping by Neural Networks,“ Neural Networks, Vol.2, pp. 183-192.

[19] Mach, M. 2009. “Evolučné algoritmy: Prvky a princípy.“ 1. vydanie, elfa s.r.o., Košice, pp. 250, ISBN 978-80-8086-123-0

[20] Pedersen, M.E.H. 2010. “Tuning & Simplifying Heuristical Optimization,“ PhD Thesis, University of Southampton, School of Engineering Sciences, Computational Engineering and Design Group.

[21] Clerc, M. 2012. “Standard Particle Swarm Optimisation,“ HAL open access archive.

[22] Vaščák, J. 2008. “Fuzzy logika v regulácii. Podrobný úvod do problematiky fuzzy riadenia,“ TUKE, FEI, KKUI.

[23] Mamdani, E. H. 1974. “Application of fuzzy algorithms for the control of a simple dynamic plant.“ In Proc IEEE, pp. 121-159.

[24] Sugeno, M. 1985. “An Introductory Survey of Fuzzy Control, Information Sciences,“ N.36, pp. 59-83.

[25] Takagi, T., Sugeno, M. 1985. “Fuzzy identification of systems and its applications to modeling and control,“ IEEE Transactions on Systems Man and Cybernetics, Vol. SMC-15, pp. 116-132.

[26] Lazar, T., Madarász L., et al. 2011. “Inovatívne výstupy z transformovaného experimentálneho pracoviska s malým prúdovým motorom,” monografia, FEI, LF, TUKE, elfa, s.r.o., Košice, ISBN 978-80-8086-170-4, EAN 978 8080 861704

[27] Hladký, V. 2005. “A Contribution to Hybrid Control.“ Symposium on Applied Machine Intelligence and Informatics (SAMI 2005), pp. 195-201, ISBN 963-7154-35-3.

[28] Lin, C.-T. , Lee, C. S. G. 1996. “Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems,“ Prentice-Hall PTR.

[29] Sher, G. I. 2012. “Handbook of Neuroevolution Through Erlang.“ Springer Verlag, pp. 831, ISBN 978-1-4614-4463-3.

[30] Kassahun, Y., Edgington, M., Metzen, J. H., Sommer, G., Kirchner, F. 2007. “Common Genetic Encoding for Both Direct and Indirect Encodings of Networks,“ Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007), London, UK, pp. 1029-1036.

[31] Stavrakoudis, D. G., Galidaki G. N., Gitas I. Z., Theocharis J. B. 2010. “Enhancing the Interpretability of Genetic Fuzzy Classifiers in Land Cover Classification from Hyperspectral Satellite Imagery,“ WCCI 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain.

[32] Freund, Y., Schapire R. 1996. “Experiments with a new boosting algorithm,“ Proceedings of 13th International Conference on Machine Learning, pp. 148–156.

[33] Hoffman F. 2004. “Combining boosting and evolutionary algorithms for learning of fuzzy classification rules,“ Fuzzy Sets Syst., vol. 14, no. 1, pp. 47–58.

[34] Beneš, J. 1984. “Některé otázky syntézy kooperatívního, evolučního a situačního řízení.“ Automatizace , roč. 27, č. 8-9, pp. 213-216.

[35] Beneš, J. 1983. “Nové výhledy technické kybernetiky.“ Věda a lidstvo. Horizont, Praha, pp. 241-252.

[36] Papageorgiou, E. I., Salmeron, J. L. 2013. “A Review of Fuzzy Cognitive Maps research during the last decade,“ IEEE Transactions on Fuzzy Systems, vol. 21, no. 1, pp. 66-79.

[37] León, M., Rodriguez, C., García, M.M., Bello, R., Vanhoof, K. 2010. “Fuzzy Cognitive Maps for Modeling Complex Systems”. Advances in Artificial Intelligence. 9th Mexican International Conference on Artificial Intelligence, MICAI 2010, Pachuca, Mexico. Proceedings, Part I, pp. 166-174. Springer Berlin Heidelberg. ISBN 978-3-642-16760-7

[38] Li, X. 2000. “Dynamic Knowledge Inference and Learning under Adaptive Fuzzy Petri Net Framework,“ IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, vol.30, no.4, pp. 442-450. ISSN 1094-6977

[39] Axelrod, R. 1976. “Structure of decision: The Cognitive Maps of political elites,“ Princeton, N.J.: Princeton University Press.

[40] Vaščák, J., Madarász L. 2010. “Adaptation of Fuzzy Cognitive Maps – a Comparison Study,“ Acta Polytechnica Hungarica, Vol. 7, No. 3, pp. 109-122.

[41] Gregor, Michal., Groumpos, P. P. 2013. “Tuning the Position of a Fuzzy Cognitive Map Attractor Using Backpropagation through Time“, The 7th International Conference on Integrated Modeling and Analysis in Applied Control and Automation, Athens, Greece.

[42] Kosko B. 1986. “Fuzzy Cognitive Maps“, International Journal of Man-Machine Studies, Elsevier, Vol. 24, No. 1, pp. 65-75.

[43] Papageorgiou, E.I. 2012. “Learning Algorithms for Fuzzy Cognitive Maps – A Review Study,“ IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol.42, no.2, pp. 150-163.

[44] Ketipi, M.K., Koulouriotis, D.E., Karakasis, E.G., Papakostas, G.A., Tourassis, V.D. 2011. “Nonlinear cause-effect relationships in Fuzzy Cognitive Maps,“ 2011 IEEE International Conference on Fuzzy Systems (FUZZ), pp. 836-843.

[45] Chen S. M. 1995. “Cognitive-Map-based Decision Analysis Based on NPN Logics“, Fuzzy Sets and Systems, Elsevier, Vol. 71, No. 2, pp. 155-163.

[46] Gregor, Michal., Groumpos, P. P. 2013. “Training Fuzzy Cognitive Maps Using Gradient-Based Supervised Learning“, IFIP Advances in Information and Communication Technology, Volume 412, pp. 547-556.

[47] Vaščák, J. 2013. “Príklady aplikácií fuzzy kognitívnych máp v robotike,“ Sborník příspěvků z letní školy Mezioborové přístupy informatiky a kognitivní vědy. Hradec Králové. Gaudeamus. pp. 74-92. ISBN 978-80-7435-311-6

[48] Stylios, Ch. D., Groumpos, P. P. 2004. “Modeling Complex Systems Using Fuzzy Cognitive Maps,“ IEEE Transactions on Systems Man and Cybernetics, Part A: Systems and Humans, vol. 34, no. 1.

[49] Stach, W., Kurgan, L., Pedrycz, W. 2008. “Data-driven Nonlinear Hebbian Learning method for Fuzzy Cognitive Maps,“ IEEE International Conference on Fuzzy Systems (FUZZ), IEEE World Congress on Computational Intelligence. pp. 1975-1981.

[50] Diesinger, A.L. 2008. “Systems Of Commercial Turbofan Engines,“ Springer-Verlag, pp. 34, ISBN 978-3-540-73618-9

[51] Baskharone, E. A. 2006 “Principles of Turbomachinery in Air-Breathing Engines,“ Cambridge Aerospace Series, Cambridge University Press, pp. 600, ISBN-13: 978-0521858106.

[52] Lazar, T., et. al. 2000. “Tendencie vývoja a modelovania avionických systémov,“ 160 pp, ISBN 80-88842-26-3, MoSR, Bratislava.

[53] Főző, L., Andoga, R., Madarász, L., Kolesár, J., Judičák J. 2015. “Description of an Intelligent Small Turbocompressor Engine with Variable Exhaust Nozzle,“ IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI), January 22-24, 2015, Herl’any, Slovakia. ISBN 978-1-4799-8221-9.

[54] Hagiwara, M. 1992. “Extended fuzzy cognitive maps,“ IEEE International Conference on Fuzzy Systems, pp.795,801, 8-12 Mar 1992.

[55] Puheim, M., Vaščák J., Madarász L. 2014. “Three-Term Relation Neuro-Fuzzy Cognitive Maps.” In: CINTI 2014 : 15th IEEE International Symposium on Computational Intelligence and Informatics : Proceedings : November 19-21, 2014, Budapest. - Danvers : IEEE, 2014 P. 477-482. - ISBN 978-1-4799-5337-0

[56] Puheim, M., Nyulászi, L., Madarász, L., Gašpar, V. 2014. “On Practical Constraints of Approximation Using Neural Networks on Current Digital Computers.” IEEE 18th International Conference on Intelligent Engineering Systems : Proc., Tihany, Hungary, 3-5 July 2014, pp. 257-262, ISBN 978-1-4799-4616-7.

[57] Bennett, S. 1993. “A history of control engineering, 1930-1955,” IEE Control Engineering Series 47, ISBN 978-0-86341-299-8.

[58] Takagi, H. 2001. “Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation,” Proceedings of the IEEE, vol. 89., iss. 9, pp. 1275-1296.

[59] Mls, K., Cimler, R., Vaščák, J., Puheim, M. 2017. “Interactive evolutionary optimization of fuzzy cognitive maps.” In: Neurocomputing. Vol. 232, p. 58-68. ISSN 0925-2312

[60] Puheim, M. 2015. “pyRoSim - Python Robotic Simulator.” GitHub source code repository. [Online]. Available: https://github.com/mpuheim/pyRoSim [Acessed: 7-August-2018].

[61] Takagi, H. 2012. “Interactive evolutionary computation for analyzing human awareness mechanisms.” Appl. Comp. Intell. Soft Comput. 2012. pp. 1-12.

[62] Farooq, H., Siddique, M. T. 2014. “A comparative study on user interfaces of interactive genetic algorithm,” Procedia Computer Science, vol. 32, pp. 45–52. The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014).

[63] Zelinka, I. 2016. “SOMA-Self-organizing Migrating Algorithm,” Studies in Computational Intelligence, Springer-Verlag, Berlin, vol. 626, pp. 3–49. doi:10.1007/978-3-319-28161-2_1

[64] Puheim, M., Vaščák, J. 2017. “Semi-Interactive Population-Based Optimization of Fuzzy Cognitive Maps”. Unpublished manuscript.

[65] Statista. 2018. “Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025.” [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide. [Accessed: 13-June-2018].

[66] Friess, P. 2013. “Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems,” River Publishers, ISBN 9788792982735

[67] Augusto, J. C., Callaghan, V., Cook, D., Kameas, A., Satoh, I. 2013. “Intelligent Environments: a manifesto,” In: Human-centric Comp. and Information Sciences, 2013, doi:10.1186/2192-1962-3-12

[68] Zhang, D., Ning, H., Xu, K. S., Lin, F., Yang, L. T. 2012. “Internet of Things.” J. UCS 18 : 1069-1071. doi: 10.1007/978-3-642-32427-7

[69] Fisher, M. 2017. “Evolution of the Internet of Things!” Infographics. [Online]. Available: https://twitter.com/fisher85m/status/926360908900773889. [Accessed: 23-June-2018].

[70] FS.COM. 2017. “TCP/IP vs. OSI: What’s the Difference Between the Two Models?” [Online]. Available: https://community.fs.com/blog/tcpip-vs-osi-whats-the-difference-between-the-two-models.html. [Accessed: 23-June-2018].

[71] Čurová, D., Haluška, R., Hugec, T., Puheim M., Vaščák J., Sinčák, P. 2017. “Intelligent space at center for intelligent technologies – system proposal.” In: SAMI 2017. - Danvers : IEEE, 2017 S. 191-195. - ISBN 978-1-5090-5654-5

[72] Hvizdoš, J., Vojtko, M., Koscelanský, M., Haluška R., Pavlov J. 2017. “Applications of Remote Controlled Robotics in the Intelligent Space.” In: SAMI 2017. - Danvers : IEEE, 2017 S. 117-121. - ISBN 978-1-5090-5654-5

[73] Puheim M., Hvizdoš J., Szabóová M., Vaščák J. 2017. “Aplikácie vzdialene ovládanej robotiky v inteligentnom priestore (1),” ATP Journal, 2017, Vol. 24, No. 9, pp. 53-55, ISSN 1335-2237.

[74] Atzori, L. et al. 2010. “The Internet of Things: A survey.” In: Computer networks, Vol. 54, No. 15, pp. 2 787 – 2 805. doi: 10.1016/j.comnet.2010.05.010

[75] Vermesac, C., Friess, P. 2011. “Internet of Things – Global Technological and Societal Trends.” Denmark: River Publishers. 316 pp. ISBN 978-87-92329-67-7.

[76] Miorandi, D., Sicari, S., Pellegrini, F., Chlamtac, I. 2012. “Internet of things: Vision, application and research challenges.” Ad Hoc Networks, vol. 10, iss. 7, pp. 1497–1516. DOI: https://doi.org/10.1016/j.adhoc.2012.02.016

[77] Johnston S., 2009. “Diagram showing overview of cloud computing, with typical types of applications supported by that computing model.” Wikimedia Commons. Available at: https://en.wikipedia.org/wiki/File:Cloud_computing.svg.

[78] ISO/IEC 7498-1:1994. “Open Systems Interconnection -- Basic Reference Model: The Basic Model.”

[79] Microsoft. 2017. “Windows Network Architecture and the OSI Model.” [Online]. Available: https://docs.microsoft.com/en-us/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model [Accessed: 23-June-2018]

[80] Kozierok, Ch. M. 2005. “The TCP/IP Guide.” [Online]. Available: www.tcpipguide.com [Accessed: 26-June-2018]

[81] Mell, P. and Grance, T., 2010. “The NIST definition of cloud computing.” Communications of the ACM, vol. 53, iss. 6, p. 50

[82] Mazikglobal. 2018. “What is cloud computing stack (SaaS, PaaS, IaaS).” [Online]. Available: http://www.mazikglobal.com/blog/cloud-computing-stack-saas-paas-iaas/ [Accessed: 16-July-2018]

[83] WinSystems. 2018. “Cloud, Fog and Edge computing: What’s the Difference?” [Online]. Available: https://www.winsystems.com/cloud-fog-and-edge-computing-whats-the-difference/ [Accessed: 20-July-2018]

[84] Singh, N. K. 2015. “Seminar Report on Fog Computing,” Cochin University of Science and Technology, August 2015.

[85] i-SCOOP. 2018. “Fog computing: fog and cloud along the Cloud-to-Thing continuum.” [Online]. Available: https://www.i-scoop.eu/internet-of-things-guide/fog-computing-cloud-internet-things/ [Accessed: 21-July-2018]

[86] Iorga, M., Feldman, L., Barton, R., Martin, M. J., Goren, N., Mahmoudi, Ch. 2018.” Fog Computing Conceptual Model.” NIST Special Publication 500-325. Recommendations of the National Institute of Standards and Technology. March 2018. Available: https://doi.org/10.6028/NIST.SP.500-325

[87] Center for Intelligent Technologies. 2016. "Ai-cit.sk" [Online]. Available: http://www.ai-cit.sk/. [Accessed: 30-July-2018].

[88] Hvizdoš, J. 2018. “Ubiquitous Robotics in Intelligent Space: New Approaches in Navigation of Ubiquitous Robots.” PhD Thesis. Košice, FEEI, DCAI, TUKE.

[89] Kim, J.H., Lee, K.H., Kim, Y.D., Kuppuswamy, N.S. and Jo, J., 2007. “Ubiquitous robot: A new paradigm for integrated services.” In Robotics and Automation, April 2007, IEEE International Conference on Robotics and Automation, pp. 2853-2858.

[90] Vaščák, J. 2017. “Internet of Things, Robotics and Computational Intelligence: Towards the Synergy of Autonomous Systems.” Habilitation thesis. Košice, FEEI DCAI, TUKE.

[91] Di Rocco, M., Sathyakeerthy, S., Grosinger, J., Pecora, F., Saffiotti, A., Bonaccorsi, M., Cavallo, F., Limosani, R., Manzi, A., Teti, G., Dario, P. 2014. “A planner for ambient assisted living:from high-level reasoning to low-level robot execution and back.” In: AAAI Spring Symposium on Qualitative Representations for Robots, Stanford University, Palo Alto, California, USA, pp. 10–17

[92] Tarhaničová, M., Machová, K., Sinčák, P. 2015. “Computers Capable of Distinguishing Emotions in Text.” In: Advances in Intelligent Systems and Computing, Switzerland, Springer, 2015 Vol. 316, pp. 61-69, ISBN 978-3-319-10782-0, ISSN 2194-5357.

[93] Petrušová, A. 2016. “3D visualization and control of smart home using cloud platform.” Bachelor thesis. Košice, FEEI, DCAI, TUKE.

[94] Tomaľa, D. 2017. “Event Detection from Videos in the Intelligent Space.” Master thesis. Košice, FEEI, DCAI, TUKE.

[95] Haluška, R. 2017. “Image Information Processing for Needs of Intelligent Space Applications.” Master thesis. Košice, FEEI, DCAI, TUKE.

[96] Hugec, T. 2017. “Speech Processing in Intelligent Space focused on Emotion Analysis.” Bachelor thesis. Košice, FEEI, DCAI, TUKE.

[97] Pacholská, N. 2017. “Data Management for the Internet of Things.” Bachelor thesis. Košice, FEEI, DCAI, TUKE.

[98] Koscelanský, M. 2017. “Robot Localization in Intelligent Space.” Master thesis. Košice, FEEI, DCAI, TUKE.

[99] Vojtko, I. 2017. “Sensory Systems Based on RFID Technology in the Intelligent Space.” Bachelor thesis. Košice, FEEI, DCAI, TUKE.

[100] Kim, T. H., Choi, S. H. and Kim, J. H., 2007. “Incorporation of a software robot and a mobile robot using a middle layer.” Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 37, no. 6, pp. 1342-1348.

[101] GLORY LTD. 2018. “A collaboration of a facial recognition cloud service and Pepper enables assisting visitor and notification of visit by a chat system and emails.” News Release. [Online]. Available: http://glory-request.com/groupinfo/news_releases/2017/0718.html. [Accessed: 30-July-2018].

[102] Bonaccorsi, M., Fiorini, L., Sathyakeerthy, S., Saffiotti, A., Cavallo, F., Dario, P. 2015. “Design of cloud robotic services for senior citizens to improve independent living in multiple environments.” In: Intelligenza Artificiale, vol. 9, no. 1, pp. 63-72. DOI: 10.3233/IA-150077

[103] Zuckenberg, M. 2016. “Building Jarvis.” Facebook Note. [Online]. Available: https://www.facebook.com/notes/mark-zuckerberg/building-jarvis/10154361492931634/. [Accessed: 2-August-2018]

[104] Puheim, M., Vaščák, J., Madarász, L. 2015. “A Proposal for Multi-Purpose Fuzzy Cognitive Maps Library for Complex System Modeling.” In IEEE 13 th International Symposium on Applied Machine Intelligence and Informatics. January 22–24, 2015. pp. 175-180. ISBN: 978-1-4799-8220-2

[105] FCMappers. 2011. “FCMappers Website.” Disconnecting the Missing Link. [Online]. Available: http://www.fcmappers.net/ [Acessed: 19-August-2018].

[106] Aspuru., G. O. 2006. “Fuzzy Cognitive Maps.” [Online]. Available: http://www.ochoadeaspuru.com/fuzcogmap/index.php [Accessed: 3-August-2018].

[107] Franciscis, D., 2014. “JFCM : A Java Library for Fuzzy Cognitive Maps,” in Fuzzy Cognitive Maps for Applied Sciences and Engineering. Intelligent Systems Reference Library. Springer Berlin Heidelberg, Vol. 54, pp. 199-220, ISBN 978-3-642-39738-7.

[108] Gray, S. A., Gray, S., Cox, L., Henly-Shepard, S. 2013. “Mental modeler: A fuzzy-logic cognitive mapping modeling tool for adaptive environmental management,” in Proceedings of the 46th International Conference on Complex Systems, Hawaii, pp. 965-973, DOI 10.1109/ HICSS.2013.399

[109] Papageorgiou, E., Pispas, N. 2018. “Fuzzy Cognitive Maps Wizard.” [Online]. Available: http://fcmwizard.com [Accessed: 15-September-2018].

[110] Papageorgiou, E. 2018. “Summer School on Fuzzy Cognitive Maps.” [Online]. Available: http://epapageorgiou.com/summer-school/ [Accessed: 15-September-2018].

[111] León, M., Rodriguez, C., García, M. M., Bello, R., Vanhoof, K. 2010. “Fuzzy Cognitive Maps for Modeling Complex Systems,” in 9th Mexican International Conference on Artificial Intelligence, MICAI. Pachuca, Mexico, pp 166-174 ISBN 978-3-642-16760-7

[112] MathWorks. 2014. “Fuzzy Logic Toolbox.” [Online]. Available: http://www.mathworks.com/products/fuzzy-logic/ [Accessed: 4-August-2018].

[113] Mono Project. 2018. “Mono - Cross platform, open source .NET framework”. [Online]. Available: https://www.mono-project.com/ [Accessed: 4-August-2018].

[114] Puheim, M. 2016. “OpenFCM – Fuzzy Cognitive Maps Library.” GitHub source code repository. [Online]. Available: https://github.com/mpuheim/OpenFCM [Accessed: 7-August-2018].

[115] Puheim, M. 2018. “PyOpenFCM - Python Open Fuzzy Cognitive Maps Library (with Web API).” GitHub source code repository. [Online]. Available: https://github.com/mpuheim/PyOpenFCM [Accessed: 4-August-2018].

[116] Puheim, M. 2016. “Current State of Control and Modeling of Complex Systems Using Fuzzy Cognitive Maps.” In: Proc. of SCYR 2016. TU Košice. pp. 86-87. ISBN 978-80-553-2566-8

[117] Puheim, M. 2017. “Control of intelligent complex systems.” In: Proc. of SCYR 2017. TU Košice. pp. 64-65. ISBN 978-80-553-3162-1

[118] Paračka, F., Puheim, M. 2016. “Graphical User Interface for OpenFCM Library.” In: Electrical Engineering and Informatics 7 : Proc. of the FEI TU, Košice. pp. 216-220. ISBN 978-80-553-2599-6

[119] Nachmanson, P. 2016. “Microsoft Automatic Graph Layout.” [Online]. Available: http://research.microsoft.com/en-us/projects/msagl/ [Accessed: 12-Mar-2016].

[120] Puheim, M. 2018. “PyIoTsim – Python IoT Simulator.” GitHub source code repository. [Online]. Available: https://github.com/mpuheim/PyIoTsim [Accessed: 4-August-2018].

[121] Medium. 2017. “Advantages and Disadvantages of Python Programming Language.” [Online]. Available: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121 [Accessed: 12-August-2018]

[122] Puheim, M. 2018. “Python Open Fuzzy Cognitive Maps Library – Documentation.” [Online]. Available: https://mpuheim.github.io/PyOpenFCM/docs/html/index.html [Accessed: 12-August-2018]

[123] Puheim, M., Vaščák, J., Machová, K. 2016. “Efficient FCM computations using sparse matrix-vector multiplication.” In: SMC 2016. Danvers:IEEE. pp. 4165-4170. ISBN 978-1-5090-1819-2

[124] Puheim, M. 2016. “Sparse Matrix Vector Multiplication Benchmark.” GitHub source code repository. [Online]. Available: https://github.com/mpuheim/SMVM-Benchmark [Accessed: 12-August-2018]

[125] Ronacher, A. 2018. “Flask.” [Online]. Available: http://flask.pocoo.org/ [Accessed: 14-August-2018]

[126] Bostock, M. 2017. “D3 Data-Driven Documents.” [Online]. Available: https://d3js.org/ [Accessed: 16-August-2018]

[127] Parsons D. 2012. “Event-Driven Programming.“ In: Foundational Java. Springer, London. ISBN 978-1-4471-2478-8

[128] Kalal, Z., Mikolajczyk, K. and Matas, J., 2012. “Tracking-learning-detection.“ IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 34, No 7, pp. 1409-1422. DOI: 10.1109/TPAMI.2011.239

[129] Puheim, M. 2018. “Situational Control Strategy for Robot Navigation.“ Youtube video. [Online]. Available: https://youtu.be/Y1oM9p-VfjE [Accessed: 2-November-2018]

Appendices

Appendix A – Resumé v slovenskom jazyku

Appendix B – References to software and documentation

Appendix C – Publications of the author

Appendix D – Author's CV

Appendix E – CD medium with software and dissertation in electronic form

Appendix A – Resumé

Riadenie zložitých systémov je v súčasnosti s ohľadom na nástup internetu vecí, inteligentných priestorov a všadeprítomnej robotiky veľmi aktuálnou problematikou. Na riešenie týchto problémov je možné úspešne využiť prostriedky umelej inteligencie, napríklad fuzzy kognitívne mapy (FCM). Implementácia stratégií pre riadenie zložitých systémov je možná použitím viacerých prístupov, napríklad metodiky situačného riadenia alebo formátorového riadenia, ktoré sú v práci použité na návrh riadenia inteligentného priestoru v situačnom rámci navigácie mobilného robota.

Základné tézy doktorskej dizertačnej práce boli určené nasledovne:

1) Modifikovať základnú koncepciu FCM s cieľom umožniť modelovanie a riadenie zložitých systémov (vedecká téza).
2) Vytvoriť programovú knižnicu podporujúcu výpočty pomocou FCM pre modelovanie a riadenie zložitých systémov a podporné grafické (GUI) a aplikačné programové rozhrania (API) (technologická téza).
3) Navrhnúť situačný model riadenia v inteligentnom priestore s využitím FCM (vedecká téza).
4) Navrhnúť štruktúru riadenia inteligentného priestoru v situačnom rámci úloh navigácie robota (vedecká téza).
5) Vytvoriť simulačný systém situačného riadenia prvkov inteligentného priestoru (technologická téza).

Prvý cieľ je splnený v kapitole 4, ktorá uvádza niekoľko modifikácií FCM, ktoré zjednodušujú modelovanie zložitých systémov, vrátane troj-termových a neurálnych relácií (kap. 4.2), spolu s učiacimi algoritmami (kap. 4.3) umožňujúcimi automatizovať adaptáciu modelov vytvorených pomocou FCM.

Druhý cieľ je naplnený v kapitole 6, ktorá prezentuje návrh a implementáciu novo-vytvorenej programovej knižnice pre tvorbu modelov FCM (kap. 6.2) vrátane webového aplikačného programového rozhrania (kap. 6.3) a odpovedajúceho grafického rozhrania (kap. 6.4).

Tretí cieľ je zrealizovaný v kapitole 7.1, kde je uvedený zovšeobecnený popis a globálne ciele riadenia inteligentného priestoru (kap. 7.1.1) spoločne s návrhom situačnej dekompozície (kap. 7.1.2) a návrhom rámcovej riadiacej štruktúry (kap. 7.1.3) založenej na formátorovom riadení pozostávajúcom z viacerých sieťových regulátoroch a analyzátoroch implementovaných pomocou FCM.

Štvrtý cieľ je splnený z prevažnej časti v kapitole 7.4, avšak partikulárne riešenia sú diskutované už v predchádzajúcich kapitolách, vrátane všeobecného popisu problematiky navigácie mobilného robota (kap. 7.2) a konkrétnej implementácie regulátorov a analyzátorov na báze FCM (kap. 7.3).

Riešenie posledného cieľa, ktorým je simulácia vybraných prvkov inteligentného priestoru nevyhnutných pre realizáciu úlohy navigácie mobilného robota, je uvedené v kapitole 7.5.

Pri riadení zložitých systémov (kap. 1), ktoré obsahujú veľký počet prvkov (technických, netechnických prostriedkov a ľudí), veľmi často nedokážeme vytvoriť presný matematický model riadeného systému. V prípade, že by sa nám ho aj podarilo zostrojiť, bol by pravdepodobne veľmi zložitý a pri riadení ťažko prakticky využiteľný. Pri návrhu riadenia takéhoto systému však môžeme využiť metodiku situačného riadenia (kap. 1.4), ktorá vykoná dekompozíciu riadenia daného systému na menšie problémy, závislé od aktuálneho stavu systému, ktoré už je možné vyriešiť.

Návrh systému situačného riadenia pozostáva z viacerých krokov. Etapa plánovania situačného riadenia zložitého systému začína popisom riadeného systému, určením globálnych cieľov riadenia a následne návrhom tzv. situačných tried (rámcov), ktoré môžu zoskupovať viacero podobných situácií, resp. stavov systému. K jednotlivým situačným triedam sú potom priradené príslušné stratégie riadenia. Úlohou operatívneho riadenia je parametrizácia vopred pripravených stratégií riadenia podľa vyhodnotenia stavu systému a ich použitie pri jeho riadení. Cieľom je v reálnom čase vyrovnať odchýlky od požadovaného stavu systému. Proces riadenia obsahuje tri základné rozhodovacie uzly (pozri str. 24), v ktorých podľa zvolených rozhodovacích kritérií a aktuálnej situácie prebieha rozhodovanie:

- o zatriedení do príslušnej vzorovej triedy situácií (estimácia stavu systému),
- o voľbe zodpovedajúcej stratégie,
- o realizácii čiastkových funkcií riadenia.

Keďže situačné riadenie je rámcovou metodikou, pre riešenie jednotlivých úloh, ktoré súvisia s rozhodovaním (napr. analýza, klasifikácia, modelovania atď.), je možné aplikovať metódy (kap. 2.3) z rôznych oblastí, vrátane umelej inteligencie, konkrétne aj FCM.

Umelá inteligencia sa venuje návrhu systémov, ktoré majú schopnosť riešiť úlohy o ktorých sa predpokladá, že by ich vyriešil len človek s poznatkami o danej problematike. Medzi základné vlastnosti inteligentných technológií patria schopnosti učiť sa z dát a získavať poznatky, ukladať poznatky (získané v minulosti) a následne využívať získané poznatky (pri riešení problémov v budúcnosti).

Moderné prístupy (kap. 2) založené na metódach výpočtovej inteligencie („Computational Intelligence“, „Soft Computing“) sú charakteristické nesymbolickou reprezentáciou vedomostí. Predstavujú prístup „zdola nahor“, ktorý je inšpirovaný prevažne nízkoúrovňovými percepčnými procesmi, biologickými procesmi, evolučnými princípmi a prírodnými javmi. Prevládajúcou filozofiou je tzv. konekcionizmus, ktorý predpokladá, že z jednoduchých častí je možné spájaním poskladať sofistikované systémy z vysokou abstraktnou inteligenciou. V rámci tejto práce bol uvedený prehľad troch prostriedkov výpočtovej inteligencie (kap. 2.1), konkrétne umelých neurónových sietí (kap. 2.1.1), evolučných algoritmov (kap. 2.1.2) a fuzzy inferenčných systémov (kap. 2.1.3). Kombináciou uvedených prostriedkov vznikajú hybridné inteligentné systémy (kap. 2.2.2).

Fuzzy kognitívne mapy (kap. 3) sú široko využívaným inferenčným nástrojom pre modelovanie kvalitatívnych i kvantitatívnych komplexných vzťahov, v rámci širokého spektra rôznych technických i netechnických systémov, a to jednoduchým a zrozumiteľným spôsobom. V poslednej dekáde hrali FCM významnú úlohu pri aplikáciách v mnohých oblastiach vedeckého výskumu, ako sú spoločenské a politické vedy, inžinierstvo, informačné technológie, robotika, expertné systémy, medicína, vzdelávanie, predikcia, životné prostredie a ďalších. Vďaka svojej cyklickej štruktúre sú FKM schopné popisovať komplexné dynamické systémy. Je možné skúmať napr. limitné cykly systému, kolízie atď. Azda najväčšou výhodou je ich jednoducho zrozumiteľná reprezentácia znalostí, ktorá sa dá názorne vizualizovať v grafickej podobe (pozri str. 50).

S cieľom zjednodušiť modelovanie zložitých systémov pomocou FKM (kap. 4) sme navrhli niekoľko modifikácií základnej štruktúry vzťahov (relácií) medzi konceptami, vrátane troj-termových a neurálnych relácií (kap. 4.2), spolu s učiacimi algoritmami (kap. 4.3) umožňujúcimi automatizovať adaptáciu modelov vytvorených pomocou FCM. Azda najvýznamnejším vlastným príspevkom v tejto oblasti je implementácia interaktívnej evolučnej adaptácie FKM (kap. 4.3.1), ktorá bola publikovaná v [59]. V nadväznosti na túto prácu sme vykonali aj experimenty so semi-interaktívnou evolúciou (kap. 4.3.2), ktorá kombinuje expertné a automatické hodnotenie heuristickou funkciou. Výsledky (uvedené v Tab. 3 až Tab. 6) ukazujú viaceré veľmi zaujímavé zlepšenia súvisiace s celkovým zrýchlením procesu učenia a znížením záťaže a únavy experta.

V ďalšej časti práce (kap. 5) práce sme uviedli základné poznatky z oblasti internetu vecí (kap. 5.1), inteligentných priestorov (kap. 5.2) a všadeprítomnej robotiky (kap. 5.2.4), ktoré sú potrebné pre realizáciu aplikácií a návrhu riadiacich štruktúr. V tejto súvislosti sme demonštrovali jednoznačné potvrdenie inteligentného priestoru ako zložitého systému (kap. 5.3), a preto argumentujeme možnosťou využiť všetky metódy, ktoré sú uvedené v predchádzajúcich kapitolách, vrátane

situačného riadenia (kap. 1.4) a formátorového riadenia (kap. 2.3). V rámci úvah v oblasti internetu vecí a inteligentného priestoru, sme diskutovali o základných technológiách, ktoré umožňujú ich implementáciu, vrátane základov modernej sieťovej komunikácie (kap. 5.1.1) a jej súvisiacich výpočtových modelov (kap. 5.1.2), ako je cloud, fog a edge computing. Bolo spomenutých viacero možných aplikácií internetu vecí (kap. 5.1.3) a inteligentných priestorov (5.2.3) spolu s konkrétnym návrhom inteligentného priestoru a jeho podrobným opisom (kap. 5.2.2).

S ohľadom na zistenia v predchádzajúcich kapitolách sme prezentovali softvérové nástroje (kap. 6) potrebné na implementáciu riadiacich štruktúr na báze FCM v podmienkach internetu vecí a inteligentných priestorov. Po uvedení prehľadu existujúcich nástrojov (kap. 6.1) sme navrhli a implementovali novú programovú knižnicu pre výpočty a modelovanie s využitím FCM (kap. 6.2). Možnosť jej nasadenia v prostredí internetu sme zabezpečili vytvorením odpovedajúceho aplikačného programového rozhrania (kap. 6.3) dostupného prostredníctvom metód webového protokolu HTTP. Pomocou tohto webového rozhrania sme vytvorili aj používateľské rozhrania (kap. 6.4), vrátane interaktívnej grafickej vizualizácie vytvorených FCM modelov (pozri str. 117). Ďalšie informácie o všetkých vytvorených programových prostriedkoch, vrátane odkazov na online dokumentáciu, sú dostupné v Prílohe B.

Vytvorené softvérové nástroje sme použili pri návrhu situačného riadenia inteligentného priestoru v úlohe navigácie mobilného robota (kap. 7). Po dôkladnej analýze rôznych možností použitia inteligentného priestoru (uvedeného v kap. 5.2.2), so zreteľom na príklady viacerých aplikácií (uvedených v predchádzajúcich častiach, najmä v kap. 5.2.3 a kap. 5.2.4), sme vytvorili návrh situačného riadenia (kap. 7.1), vrátane definície globálnych cieľov riadenia (kap. 7.1.1), návrhu situačnej dekompozície (kap. 7.1.2) a návrhu rámcovej štruktúry operatívneho riadenia (kap. 7.1.3) implementovanej v podobe formátorového riadenia (podľa kap. 2.3), ktoré v našom prípade pozostáva z množiny viacerých spolupracujúcich prvkov, vrátane analyzátorov stavu systému, regulátorov a ďalších programov. Pri implementácii riadiacej štruktúry predpokladáme implementáciu jednotlivých prvkov vo forme distribuovaných, vzájomne spolupracujúcich, sieťových služieb. Na implementovanie vybraných analyzátorov a regulátora mobilného robota (kap. 7.3) sme použili online modely FCM vytvorené pomocou softvérových prostriedkov predstavených v predchádzajúcej časti. S predpokladom ich použitia sme navrhli riadiacu stratégiu (kap. 7.4) použiteľnú v situačnom rámci navigácie mobilného robota. S cieľom demonštrovať funkčnosť navrhnutej riadiacej stratégie sme implementovali simuláciu inteligentného priestoru (kap. 7.5) s ôsmimi kamerami, mobilným robotom, príslušnými analyzátormi a regulátorom. Simulácia preukázala funkčnosť navrhnutého riešenia.

S ohľadom na vyššie uvedené, môžeme konštatovať nasledujúce hlavné prínosy dizertačnej práce:

- Úspešná simulácia inteligentného priestoru potvrdila aplikovateľnosť fuzzy kognitívnych máp pri modelovaní a riadení zložitých systémov.
- Vytvorením špecializovanej knižnice sme dosiahli výrazné uľahčenie návrhu, implementácie a aplikačného nasadenia modelov fuzzy kognitívnych máp. Okrem samotných programov je zjednodušenie podporené aj možnosťou použitia troj-termových a neurálnych relácií medzi konceptami.
- Demonštrovali sme uskutočniteľnosť automatizovanej adaptácie fuzzy kognitívnych máp využitím metód spätného šírenia chyby a evolučnej optimalizácie.
- A nakoniec sme dokázali aplikovateľnosť fuzzy kognitívnych máp v úlohe sieťových regulátorov implementovaných vo forme fog alebo cloudových služieb.

V ďalšom výskume budeme hľadať nové možnosti vylepšenia adaptačných algoritmov pre fuzzy kognitívne mapy, optimalizáciu rýchlosti učenia a zvyšovanie presnosti vytvorených modelov. Taktiež je možné zlepšiť výkon sieťových regulátorov, napríklad optimalizáciou hierarchickej štruktúry sieťových zariadení a príslušných služieb zúčastňujúcich sa na formátorovom riadení. Cieľom je znižovanie latencie a množstva údajov prenášaných prostredníctvom siete. Prirodzene, najvyšším cieľom je implementovať navrhnuté algoritmy v skutočných podmienkach, prípadne sa pokúsiť tieto myšlienky komercionalizovať.

Appendix B – References to software and documentation

Several software programs were developed by the author in order to support pre-requisite research and complete the objectives of this dissertation. Most notable programs and libraries include:

- ***pyRoSim*** – *Python Robotic Simulator*
  - repository [60]: https://github.com/mpuheim/pyRoSim
  - used in publication [59]:
    - Mls, K., Cimler, R., Vaščák, J., Puheim, M. 2017. "Interactive evolutionary optimization of fuzzy cognitive maps." In: Neurocomputing. Vol. 232, p. 58-68. ISSN 0925-2312
- ***SMVM*** – *Sparse Matrix-Vector Multiplication Benchmark*
  - repository [124]: https://github.com/mpuheim/SMVM-Benchmark
  - used in publication [123]:
    - Puheim, M., Vaščák, J., Machová, K. 2016. "Efficient FCM computations using sparse matrix-vector multiplication." In: SMC 2016. Danvers:IEEE. pp. 4165-4170. ISBN 978-1-5090-1819-2
- ***OpenFCM*** – *.NET based Fuzzy Cognitive Maps Library*
  - repository [114]: https://github.com/mpuheim/OpenFCM
  - used in publication [117]:
    - Puheim, M. 2017. "Control of intelligent complex systems." In: Proc. of SCYR 2017. TU Košice. pp. 64-65. ISBN 978-80-553-3162-1
- ***PyOpenFCM*** – *Python Open Fuzzy Cognitive Maps Library*
  - repository [115]: https://github.com/mpuheim/PyOpenFCM
  - documentation [122]: https://mpuheim.github.io/PyOpenFCM/docs/html/
  - used for the dissertation.
- ***PyIoTSim*** – *Python IoT Simulator*
  - repository [120]: https://github.com/mpuheim/PyIoTsim
  - video demo [129]: https://youtu.be/Y1oM9p-VfjE
  - used for the dissertation.

Note, that in addition to download links, all of the software is also available on CD appended to the printed version of the dissertation (as Appendix E). In the following lines we provide a short description of each piece of software and provide basic instructions on how to use it:

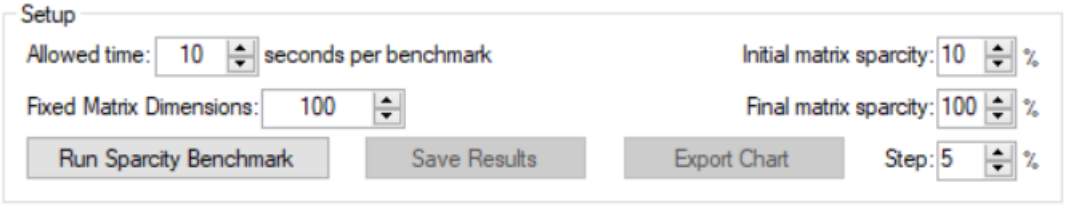
Python Robotic Simulator can be used to train an FCM-based controller of mobile robot on 2D race track using semi-interactive evolutionary algorithm:

- Dependencies:
  - *Python 3.6* and later.
  - *Pygame 1.9.3* and later.
- Download from:
  - https://github.com/mpuheim/pyRoSim/archive/master.zip
- Setup and installation:
  - Install *Python 3.*
  - From command line run: *pip install pygame*
- Execution and controls:
  - Run *interactiveEvolution.py*
  - Right mouse click to decrease fitness to 2/3 of former value (lower fitness means better individual).
  - Left mouse click to increase fitness to 3/2 of former value.
  - Left and right arrow keys to view whole generation.
- Configuration:
  - Edit *interactiveEvolution.py* to change evolution parameters.
  - Edit *modules/fcm.py* to redefine FCM controller. See *docs/* for details.
  - Use *tracks/_track_designer.zip* to create new tracks.
  - Run *runSimulation.py* to watch simulation of a single individual.
  - Run *viewTrajectory.py* to see a trajectory of a single individual.

Sparse Matrix-Vector Multiplication Benchmark compares speed of *Matrix-Vector Multiplication* between several best-known multiplication methods, including *Dense Matrix-Vector Multiplication* with a matrix stored either as *2D* or *jagged array*, *Compressed Row Storage*, *Compressed Column Storage* and *Incidence List Representation*. The program can be used to verify performance of *Incidence List Representation* which was used as basic implementation in all of the FCM-related software from this list:

- Dependencies:
  - *Visual Studio 2015* or later (to build the solution).
  - *.NET Framework 4.5.2* (to run the application).
- Download from:
  - https://github.com/mpuheim/SMVM-Benchmark/archive/master.zip

- Setup and installation:
  - Open *the solution* using Visual Studio.
  - Press F6 or run *Build->Build Solution* from the menu.
  - Or simply run the *SparseVectorMatrixMultiply.exe* (if already compiled).
- Execution and controls:
  - Run *SparseMatrixVectorMultiply\bin\Debug\SparseVectorMatrixMultiply.exe*
  - Open either *Sparsity Benchmarks* or *Size Benchmarks* tab.
  - Click the button to run the benchmark.
  - Click buttons to export results and charts.
- Configuration:
  - Use the controls provided with the application:



.NET based Fuzzy Cognitive Maps Library can be used on Windows OS to create FCM models. Note that the library is not fully completed and misses several features that are implemented in its Python version:

- Dependencies:
  - *Visual Studio 2015* or later (to build the solution).
  - *.NET Framework 4.5.2* (to use the library).
- Download from:
  - https://github.com/mpuheim/OpenFCM/archive/master.zip
- Setup and installation:
  - Open *the solution* using Visual Studio.
  - Press F6 or run *Build->Build Solution* from the menu.
  - Or simply use the *libfcm.dll* (if already compiled).
- Execution and controls:
  - From MATLAB command line run: *NET.addAssembly('\libfcm.dll')*
  - Create a map using command: *map = libfcm.FCM*
  - Proceed with other commands such as: *map.add("C1")*
  - Available commands can be found in: *library/BaseClasses/FCM.cs*

Python Open Fuzzy Cognitive Maps Library is probably the most important piece of software created alongside other work. It allows rapid prototyping of FCM models that can be exposed online as services using included web API. Graphical visualization of created models is available as well:

- Dependencies:
    - Python 3.6 and later,
    - pip, git, flask, jsonpickle.
- Download from:
    - https://github.com/mpuheim/PyOpenFCM/archive/master.zip
- Fast install & run Web API on Windows:
    - Install GIT from https://git-scm.com/download/win
    - Download & extract repository ZIP archive
    - Run *setup.bat* as admin.
    - Open CMD and run *fcmapi_service*
    - Open web browser at: http://localhost:5000/
- Install on other systems:
    - Install GIT.
    - Download & extract repository ZIP archive.
    - Run elevated CMD in extracted directory.
        - *pip install git+https://github.com/jsonpickle/jsonpickle.git*
        - *pip install .*
- Enable of Web API from command line:
    - *set FLASK_APP=fcmapi_app.py*
    - *set FLASK_DEBUG=0*
    - *flask run --host=0.0.0.0*
    - Open web browser at: http://0.0.0.0:5000/
- Usage of the library from Python Interpreter:
    - *from fcmlib import FCM*
    - *map=FCM()*
- Documentation
    - Index at: https://mpuheim.github.io/PyOpenFCM/docs/html/
    - Specific object documentation: FCM object, Concept object, Relations, Functions

Python IoT Simulator provides a simple demonstration of capabilities of the *PyOpenFCM* library in a simulated IoT environment. Implemented scenario includes a navigation of a mobile robot in a camera monitored intelligent space:

- Dependencies:
  - *Python 3.6* or later
  - *pygame, requests-futures*
  - *fcmlib* (i.e. *PyOpenFCM*)
- Download from:
  - https://github.com/mpuheim/PyIoTsim/archive/master.zip
- Setup and installation:
  - From command line run: *pip install pygame, request-futures*
  - Install *PyOpenFCM* using the instructions above.
- Start the simulation:
  - *onlinesim.py* - run simulation online using *fcmlib webAPI*.
- Configuration scripts:
  - *setupmap.py* - create an arbitrary new environment with cameras.
  - *simulation.py* - run offline simulation with simple rule-based controller.
  - *traincams.py* - train camera models implemented as neuro-fuzzy relations.
  - *traincontroller.py* - train robot controller implemented as three-term relations.

Appendix C – Publications of the author

Technická univerzita v Košiciach
Prehľad publikačnej činnosti
Autor: PUHEIM, Michal
Dátum generovania výstupu: 29. 10. 2018, 9:39:54

Skupina A2 - Ostatné knižné publikácie (ACA, ACB, BAA, BAB, BCB, BCI, EAI, CAA, CAB, EAJ, FAI)
Počet záznamov: 1
ACB - Vysokoškolské učebnice vydané v domácich vydavateľstvách (1)
Skupina B - Publikácie v karentovaných vedeckých časopisoch a autorské osvedčenia, patenty a objavy (ADC, ADD, AEG, AEH, BDC, BDD, CDC, CDD, AGJ)
Počet záznamov: 1
ADC - Vedecké práce v zahraničných karentovaných časopisoch (1)
Skupina C - Ostatné recenzované publikácie (ACC, ACD, ADE, ADF, AEC, AED, AFA, AFB, AFC, AFD, AFE, AFF, AFG, AFH, BBA, BBB, BCK, BDA, BDB, BDE, BDF, BEC, BED, BFA, BFB, BGH, CDE, CDF)
Počet záznamov: 23
ADF - Vedecké práce v domácich nekarentovaných časopisoch (3)
AEC - Vedecké práce v zahraničných recenzovaných vedeckých zborníkoch, monografiách (1)
AED - Vedecké práce v domácich recenzovaných vedeckých zborníkoch, monografiách (2)
AFC - Publikované príspevky na zahraničných vedeckých konferenciách (6)
AFD - Publikované príspevky na domácich vedeckých konferenciách (11)
Skupina D - Ostatné - mimo kategórií MŠSR
Počet záznamov: 1
AFL - Postery z domácich konferencií (1)

Počet záznamov spolu: 26

Menný zoznam publikácií:

ACB - Vysokoškolské učebnice vydané v domácich vydavateľstvách(1)

ACB001 [168916] **Rozhodovanie a zložitosť** / Vladimír Gašpar ... [et al.] - 1. preprac. vyd - Košice : Univerzitná knižnica TU - 2016. - 215 s. [CD-ROM]. - ISBN 978-80-553-2523-1. [GAŠPAR, Vladimír - OCELÍKOVÁ, Eva - NYULÁSZI, Ladislav - PUHEIM, Michal - ANDOGA, Rudolf]

ADC - Vedecké práce v zahraničných karentovaných časopisoch(1)

ADC001 [179377] **Interactive evolutionary optimization of fuzzy cognitive maps** / Karel Mls ... [et al] - 2017.In: Neurocomputing. Vol. 232 (2017), p. 58-68. - ISSN 0925-2312 [MLS, Karel - CIMLER, Richard - VAŠČÁK, Ján - PUHEIM, Michal]

ADF - Vedecké práce v domácich nekarentovaných časopisoch(3)

ADF001 [183782] **Inteligentný priestor v centre inteligentných technológií – návrh systému** / Michal Puheim ... [et al.] - 2017.In: ATP Journal. Č. 8 (2017), s. 36-39. - ISSN 1335-2237 [PUHEIM, Michal - HVIZDOŠ, Jakub - SZABÓOVÁ, Martina - VAŠČÁK, Ján]

ADF002 [188793] **Aplikácie vzdialene ovládanej robotiky v inteligentnom priestore (1)** / Michal Puheim ... [et al.] - 2017.In: ATP Journal. Roč. 24, č. 9 (2017), s. 53-55. - ISSN 1335-2237 [PUHEIM, Michal - HVIZDOŠ, Jakub - SZABÓOVÁ, Martina - VAŠČÁK, Ján]

ADF003 [188797] **Aplikácie vzdialene ovládanej robotiky v inteligentnom priestore (2)** / Michal Puheim ... [et al.] - 2017.In: ATP Journal. Roč. 24, č. 10 (2017), s. 44-45. - ISSN 1335-2237
[PUHEIM, Michal - HVIZDOŠ, Jakub - SZABÓOVÁ, Martina - VAŠČÁK, Ján]

AEC - Vedecké práce v zahraničných recenzovaných vedeckých zborníkoch, monografiách(1)

AEC001 [149436] **Application of Tracking-Learning-Detection for Object Tracking in Stereoscopic Images** / Michal Puheim ... [et al.] - 2015.In: Advances in Intelligent Systems and Computing vol. 316 : Emergent Trends in Robotics and Intelligent Systems. - Cham : Springer, 2015 Vol. 316 (2015), p. 323-331. - ISBN 978-3-319-10782-2 - ISSN 2194-5357
[PUHEIM, Michal - BUNDZEL, Marek - SINČÁK, Peter - MADARÁSZ, Ladislav]

AED - Vedecké práce v domácich recenzovaných vedeckých zborníkoch, monografiách(2)

AED001 [138385] **Aplikácia metódy TLD na sledovanie objektov v stereovíznom obraze** / Michal Puheim, Marek Bundzel - 2013.In: Electrical Engineering and Informatics 4 : Proceedings of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. - Košice : TU, 2013 S. 136-141. - ISBN 978-80-553-1440-2
[PUHEIM, Michal - BUNDZEL, Marek]

AED002 [173122] **Graphical User Interface for OpenFCM Library** / Filip Paračka, Michal Puheim - 2016.In: Electrical Engineering and Informatics 7 : proceedings of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. - Košice : FEI TU, 2016 S. 216-220. - ISBN 978-80-553-2599-6 Spôsob prístupu: *http://eei.fei.tuke.sk/drupal/data/EEI_VII.pdf*.
[PARAČKA, Filip - PUHEIM, Michal]

AFC - Publikované príspevky na zahraničných vedeckých konferenciách(6)

AFC001 [140886] **Forward Control of Robotic Arm Using the Information from Stereo-vision Tracking System** / Michal Puheim, Marek Bundzel, Ladislav Madarász - 2013.In: CINTI 2013 : 14th IEEE International Symposium on Computational Intelligence and Informatics : Proceedings : November 19-21, 2013, Budapest. - Piscataway : IEEE, 2013 P. 57-62. - ISBN 978-1-4799-0197-1
[PUHEIM, Michal - BUNDZEL, Marek - MADARÁSZ, Ladislav]

AFC002 [148036] **On Practical Constraints of Approximation Using Neural Networks on Current Digital Computers** / Michal Puheim ... [et al.] - 2014.In: INES 2014 : IEEE 18th International Conference on Intelligent Engineering Systems : Proceedings : July 3-5, 2014, Tihany, Hungary. - Danvers : IEEE, 2014 P. 257-262. - ISBN 978-1-4799-4616-7
[PUHEIM, Michal - NYULÁSZI, Ladislav - MADARÁSZ, Ladislav - GAŠPAR, Vladimír]

AFC003 [152843] **Three-Term Relation Neuro-Fuzzy Cognitive Maps** / Michal Puheim, Ján Vaščák, Ladislav Madarász - 2014.In: CINTI 2014 : 15th IEEE International Symposium on Computational Intelligence and Informatics : Proceedings : November 19-21, 2014, Budapest. - Danvers : IEEE, 2014 P. 477-482. - ISBN 978-1-4799-5337-0
[PUHEIM, Michal - VAŠČÁK, Ján - MADARÁSZ, Ladislav]

AFC004 [164149] **Automatic Predictor Generator and Behaviour Rule Extractor – A System Proposal** / Michal Puheim, Ján Paralič, Ladislav Madarász - 2015.In: CINTI 2015. - Danvers : IEEE, 2015 P. 155-159. - ISBN 978-1-4673-8519-0
[PUHEIM, Michal - PARALIČ, Ján - MADARÁSZ, Ladislav]

AFC005 [176675] **Efficient FCM computations using sparse matrix-vector multiplication** / Michal Puheim, Ján Vaščák, Kristína Machová - 2016.In: SMC 2016. - Danvers : IEEE, 2016 P. 004165-004170. - ISBN 978-1-5090-1819-2
[PUHEIM, Michal - VAŠČÁK, Ján - MACHOVÁ, Kristína]

AFC006 [176689] **Agent-Based Cloud Computing Systems for Traffic Management** / Jan Vascak, Jakub Hvizdos, Michal Puheim - 2016.In: Intelligent Networking and Collaborative Systems (INCoS). - Danvers : IEEE, 2016 P. 73-79. - ISBN 978-1-5090-4124-4 Spôsob prístupu: *http://ieeexplore.ieee.org/document/7695152/*.
[VAŠČÁK, Ján - HVIZDOŠ, Jakub - PUHEIM, Michal]

AFD - Publikované príspevky na domácich vedeckých konferenciách(11)

AFD001 [145208] **Diagnostics of complex systems using thermography** / František Adamčík ... [et al.] - 2014.In: SAMI 2014 : IEEE 12th International Symposium on Applied Machine Intelligence and Informatics : proceedings : January 23-25, 2014, Herľany, Slovakia. - Danvers : IEEE, 2014 S. 109-113. - ISBN 978-1-4799-3441-6
[ADAMČÍK, František ml. - BRÉDA, Róbert - LAZAR, Tobiáš - PUHEIM, Michal]

AFD002 [147223] **Introduction to Modeling of Complex Systems Using Fuzzy Cognitive Maps** / Michal Puheim - 2014.In: SCYR 2014 : 14th Scientific Conference of Young Researchers : proceedings from conference : May 20th, 2014, Herľany, Slovakia. - Košice : TU, 2014 S. 201-204. - ISBN 978-80-553-1714-4
[PUHEIM, Michal]

AFD003 [148037] **Normalization of inputs and outputs of neural network based robotic arm controller in Role of Inverse kinematic model** / Michal Puheim, Ladislav Madarász - 2014.In: SAMI 2014 : IEEE 12th International Symposium on Applied Machine Intelligence and Informatics : proceedings : January 23-25, 2014, Herľany, Slovakia. - Danvers : IEEE, 2014 S. 35-38. - ISBN 978-1-4799-3441-6
[PUHEIM, Michal - MADARÁSZ, Ladislav]

AFD004 [155968] **A Proposal for Multi-Purpose Fuzzy Cognitive Maps Library for Complex System Modeling** / Michal Puheim, Ladislav Madarász, Ján Vaščák - 2015.In: SAMI 2015. - Danvers : IEEE, 2015 S. 175-180. - ISBN 978-1-4799-8220-2
[PUHEIM, Michal - MADARÁSZ, Ladislav - VAŠČÁK, Ján]

AFD005 [159059] **Towards Control and Modeling of Complex Systems Using Fuzzy Cognitive Maps** / Michal Puheim - 2015.In: SCYR 2015. - Košice : TU, 2015 S. 320-321. - ISBN 978-80-553-2130-1
[PUHEIM, Michal]

AFD006 [167670] **IT services for analyses of various data samples** / Ján Paralič ... [et al.] - 2015.In: WIKT 2015. - Košice : TU, 2015 S. 82-86. - ISBN 978-80-553-2271-1
[PARALIČ, Ján - BABIČ, František - SARNOVSKÝ, Martin - BUTKA, Peter - HAVRILOVÁ, Cecília - MUCHOVÁ, Miroslava - PUHEIM, Michal - MIKULA, Martin - TUTOKY, Gabriel]

AFD007 [170005] **IT tools and services for analyses of different types of processes** / J. Paralič ... [et al.] - 2015.In: UVP Technikom. - Košice : Elfa, 2015 S. 27-31. - ISBN 978-80-8086-252-7
[PARALIČ, Ján - BABIČ, František - WAGNER, Jozef - TUTOKY, Gabriel - SARNOVSKÝ, Martin - GAŠPAR, Vladimír - BUTKA, Peter - BEDNÁR, Peter - PUHEIM, Michal - LUKÁČOVÁ, Alexandra - HAVRILOVÁ, Cecília - MUCHOVÁ, Miroslava - MIKULA, Martin - SMATANA, Miroslav]

AFD008 [170894] **Current State of Control and Modeling of Complex Systems Using Fuzzy Cognitive Maps** / Michal Puheim - 2016.In: SCYR 2016. - Košice : TU, 2016 S. 86-87. - ISBN 978-80-553-2566-8 Spôsob prístupu: *http://scyr.fei.tuke.sk/scyr-files/history/SCYR_2016_proceedings.pdf*.
[PUHEIM, Michal]

AFD009 [176678] **Automatické generovanie prediktorov a ich využitie pri dolovaní pravidiel** / Michal Puheim, Kristína Machová, Ján Paralič - 2016.In: WIKT and DaZ 2016. - Bratislava : STU, 2016 S. 311-315. - ISBN 978-80-227-4619-9
[PUHEIM, Michal - MACHOVÁ, Kristína - PARALIČ, Ján]

AFD010 [182022] **Intelligent space at center for intelligent technologies – system proposal** / Daniela Čurová ... [et al.] - 2017.In: SAMI 2017. - Danvers : IEEE, 2017 S. 191-195. - ISBN 978-1-5090-5654-5
[ČUROVÁ, Daniela - HALUŠKA, Renát - HUGEC, Tomáš - PUHEIM, Michal - VAŠČÁK, Ján - SINČÁK, Peter]

AFD011 [182230] **Control of intelligent complex systems** / Michal Puheim - 2017.In: SCYR 2017. - Košice : TU, 2017 S. 64-65. - ISBN 978-80-553-3162-1
[PUHEIM, Michal]

AFL - Postery z domácich konferencií(1)

AFL001 [158582] **IT tools and services for analyses of different types of processes** / J. Paralič ... [et al.] - 2014.In: University Science Park TECHNICOM. - Košice : Elfa, 2014 S. 16-16. - ISBN 979-80-8086-240-4
[PARALIČ, Ján - BABIČ, František - SARNOVSKÝ, Martin - WAGNER, Jozef - TUTOKY, Gabriel - HAVRILOVÁ, Cecília - KONCZ, Peter - GAŠPAR, Vladimír - LUKÁČOVÁ, Alexandra - PUHEIM, Michal]

Appendix D – Author's CV

europass Životopis

OSOBNÉ ÚDAJE **Puheim Michal**

Havanská 1, 040 13 Košice (Slovensko)
+421 902 845 030
puheim@gmail.com
http://lirslm.fei.tuke.sk/michal.puheim/

PRAX

2016–Súčasnosť **Učiteľ - Informatika**
Gymnázium sv. Edity Steinovej
Charkovská 1, 040 22 Košice

VZDELÁVANIE A PRÍPRAVA

2013–Súčasnosť **PhD. v odbore Kybernetika, študijný program – Inteligentné systémy**
Fakulta elektrotechniky a informatiky, TU v Košiciach

2015–2018 **Doplnkové pedagogické štúdium**
Katedra inžinierskej pedagogiky, TU v Košiciach

2011–2013 **Ing. v odbore Kybernetika, študijný program – Umelá inteligencia**
Fakulta elektrotechniky a informatiky, TU v Košiciach

2008–2013 **Bc. v odbore Kybernetika, študijný program – Inteligentné systémy**
Fakulta elektrotechniky a informatiky, TU v Košiciach

2004–2008 **Úplné stredné vzdelanie s maturitou, odbor - Gymnázium-matematika**
Gymnázium, Poštová 9, Košice

OSOBNÉ ZRUČNOSTI

Materinský jazyk: slovenčina

Cudzie jazyky

	POROZUMENIE		HOVORENIE		PÍSANIE
	Počúvanie	Čítanie	Ústna interakcia	Samostatný ústny prejav	
angličtina	B2	B2	B2	B2	B2
	Maturitná skúška z AJ (B2) Doktorandská jazyková skúška (B2) Cambridge First Certificate in English (B2)				
nemčina	A1	A2	A1	A1	A1
ruština	A1	B1	A1	A1	A1

Úrovne: A1 a A2: Používateľ základov jazyka - B1 a B2: Samostatný používateľ - C1 a C2: Skúsený používateľ
Spoločný európsky referenčný rámec pre jazyky

europass Životopis Puheim Michal

Organizačné a riadiace zručnosti

- Člen organizačného tímu Medzinárodného turnaja mladých fyzikov IYPT 2006 (International Young Physicists' Tournament) na Slovenskej Technickej Univerzite v Bratislave.
- Viceprezident a manažér marketingu študentskej spoločnosti AURORA (2006-2008).
- Člen organizačného výboru medzinárodnej vedeckej konferencie IEEE Symposium on Applied Machine Intelligence and Informatics v ročníkoch 2014, 2015 a 2016.

Pracovné zručnosti

- Absolvent študijno-výskumného pobytu "Erasmus Intensive Programme: Developing Open Source Systems Expertise in Europe", so zameraním na programátorské zručnosti, na University of Applied Sciences FH JOANNEUM (Graz-Kapfenberg, Rakúsko, 2012).

Digitálné zručnosti

SEBAHODNOTENIE				
Spracovanie informácií	Komunikácia	Vytváranie obsahu	Bezpečnosť	Riešenie problémov
Skúsený používateľ	Skúsený používateľ	Skúsený používateľ	Skúsený používateľ	Skúsený používateľ

Digitálne zručnosti - Tabuľka sebahodnotenia

Vodičský preukaz AM, B1, B

DOPLŇUJÚCE INFORMÁCIE

Záľuby Knihy, hudba, filmy, dokumentárne filmy, futbal, hokej, volejbal, plávanie, lyžovanie, história, politika, filozofia, psychológia, prírodné vedy, informatika, strategické a taktické hry, výpočtová technika a všetko okolo nej.