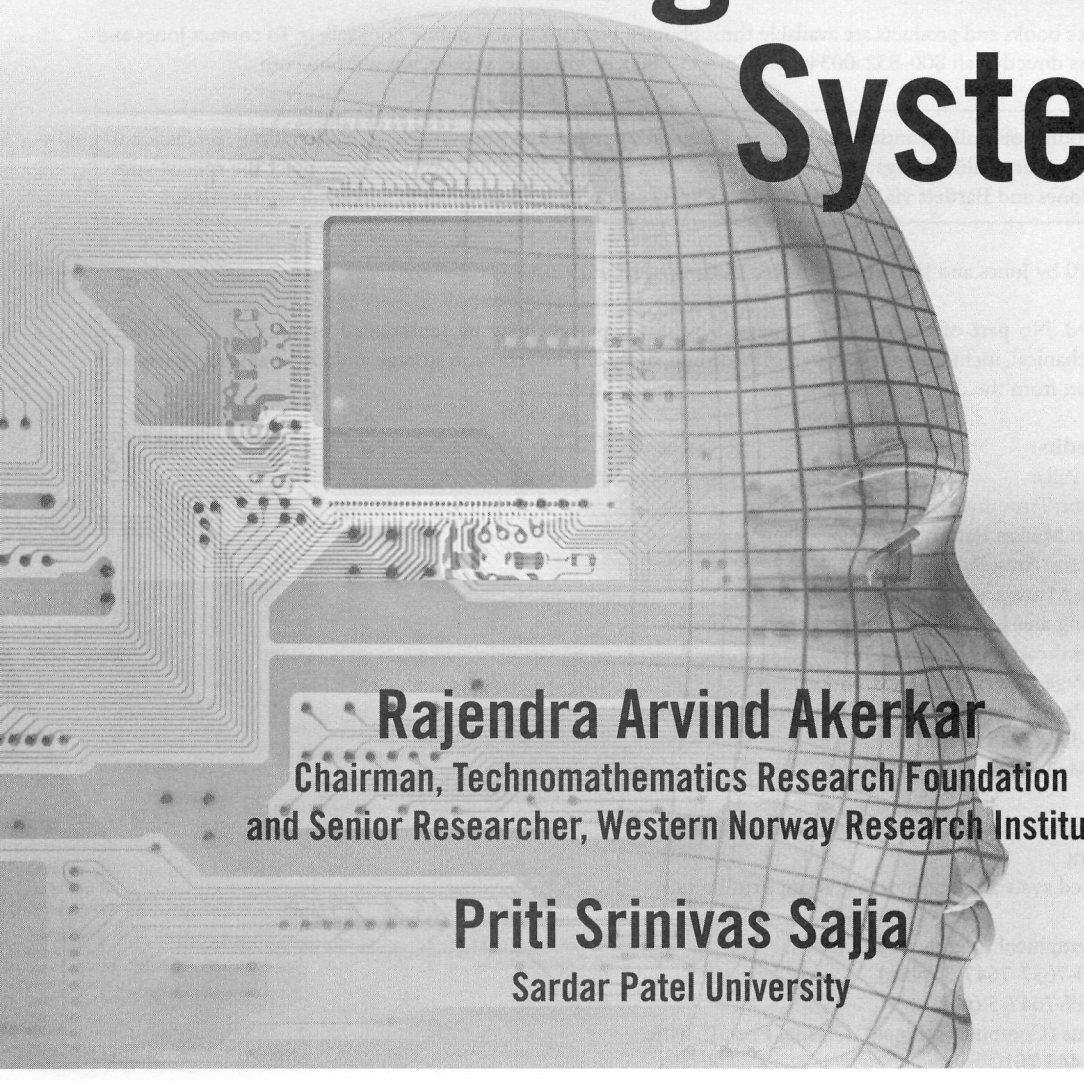
Knowledge-Based Systems



Rajendra Arvind Akerkar

Chairman, Technomathematics Research Foundation
and Senior Researcher, Western Norway Research Institute

Priti Srinivas Sajja

Sardar Patel University

JONES AND BARTLETT PUBLISHERS

Sudbury, Massachusetts

BOSTON TORONTO LONDON SINGAPORE

Contents

TMRF e-Book
Advanced Knowledge Based Systems:
Model, Applications & Research
(Eds. Sajja & Akerkar), Vol. 1, pp 1 – 11, 2010

Chapter 1

Knowledge-Based Systems for Development

Priti Srinivas Sajja, Rajendra Akerkar

INTRODUCTION

Human brain can store several thousand folds of world's knowledge. Still it is said that human brain is not fully utilized. Advances in human knowledge are tied directly to the ability to analyze to form information, process it into knowledge and communicate it to others. The human brain has approximately 1011 nerve cells called biological neurons. It is probably the most complex and least understood part of the human body. It is continuously thinking in declarative and procedural way for problem solving. But till today it is a mystery that how does the human mind work. This new millennium brought us an opportunity to attack all such questions with the help of new knowledge, new tools and new resources. Development of systems that make use of knowledge, wisdom and intelligence is a step towards meeting this challenge. The ability of the intelligent systems to capture and redistribute expertise has significant implications on development of a nation, commodity or population. Such systems allow documentation of one or more expert knowledge and utilize the knowledge for problem solving in cost effective way. It allows for, in a controlled manner, the import of expertise in various areas that the nation lacks, the export of knowledge relating to domestic areas of expertise, and the duplication and redistribution of scarce knowledge in a cost effective manner (Darek and Jain 1991). Thus areas of expertise that the selected domain/region/nation is deficient in or possesses exclusively are potential candidates of the knowledge-based systems. Though synthesized information is a key element for success, in many businesses it is a missing piece. A significant amount of Gross National Product (GNP) is invested in transferring knowledge through education and training. The AI systems effectively distribute the scarce resources for the development process. The Knowledge-Based Systems (KBS), which are a step towards an intelligent system, can be justified when a few individuals have the majority of the knowledge.

DIKW CHAIN

Data, information, knowledge and wisdom are major elements of human thinking and reasoning process. There are distinctive differences between data, information, knowledge and wisdom. Data concern with observation and raw facts. They are useless without an additional processing viz. comparing, inferring, filtering etc. The processed data is known as information. We may conclude that knowledge is a result of processes like synthesis, filtration, comparison and analysis of available information to generate meaningful outcome. Over the time, the experience, judgment, values, laws etc. are to be added to have the wisdom. This is known as Data-Information-Knowledge-Wisdom

(DIKW) chain. This chain is also known as data pyramid. These entities can be arranged as shown in Figure 1.

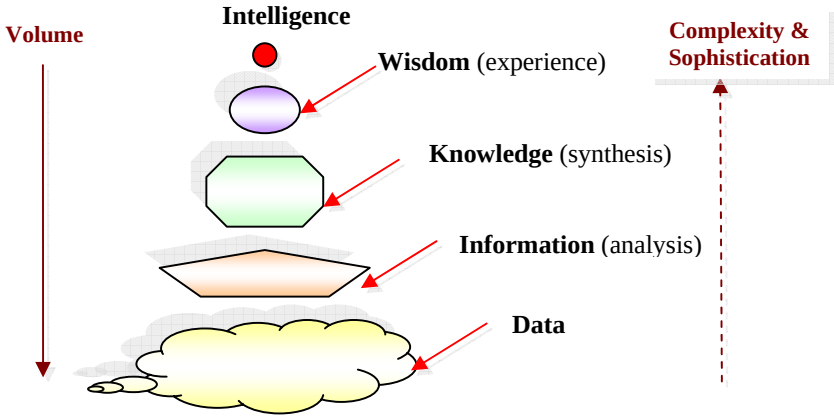


Figure 1: DIKW Chain

Knowledge can be classified in many different ways. Tacit knowledge, explicit knowledge, factual knowledge, procedural knowledge, commonsense knowledge, domain knowledge, meta knowledge, etc. Table 1 briefly introduces various types of knowledge.

Knowledge Type	Description
Domain knowledge	Domain knowledge is valid knowledge for a specified domain. Specialists and experts develop their own domain knowledge and use it for problem solving.
Meta knowledge	Meta knowledge can be defined as knowledge about knowledge.
Commonsense knowledge	Common sense knowledge is a general purpose knowledge expected to be present in every normal human being. Common-sense ideas tend to relate to events within human experience.
Heuristic knowledge	Heuristic is a specific rule-of-thumb or argument derived from experience.
Explicit knowledge	Explicit knowledge can be easily expressed in words/numbers and shared in the form of data, scientific formulae, product specifications, manuals, and universal principles. It is more formal and systematic.
Tacit knowledge	Tacit knowledge is the knowledge stored in subconscious mind of experts and not easy to document. It is highly personal and hard to formalize, and hence difficult to represent formally in system. Subjective insights, intuitions, emotions, mental models, values and actions are examples of tacit knowledge.

Table 1: Types of Knowledge

Knowledge can be represented using components like facts, rules and heuristic. Heuristic, which is a rule of thumb, can be thought as a tactic problem solving methodology, which moves solution towards success. According to J Pearl (1984), "*Heuristic in general terms are the strategies using really accessible though loosely applicable information to control problem solving process in human beings and machines*". For each problem faced in real life, there may not have exact rules and procedure for desired solution but a practically applicable rule of thumb.

KBS Structure

Knowledge-Based System (KBS) is one of the major family members of the AI group. With availability of advanced computing facilities and other resources, attention is now turning to more and more demanding tasks, which might require intelligence. The society and industry are becoming knowledge oriented and rely on different experts' decision-making ability. KBS can act as an expert on demand without wasting time, anytime and anywhere. KBS can save money by leveraging expert, allowing users to function at higher level and promoting consistency. One may consider the KBS as productive tool, having knowledge of more than one expert for long period of time. In fact, a KBS is a computer based system, which uses and generates knowledge from data, information and knowledge. These systems are capable of understanding the information under process and can take decision based on the residing information/knowledge in the system whereas the traditional computer systems do not know or understand the data/information they process.

The KBS consists of a Knowledge Base and a search program called Inference Engine (IE). The IE is a software program, which infers the knowledge available in the knowledge base. The knowledge base can be used as a repository of knowledge in various forms. This may includes an empty WorkSpace to store temporary results and information/knowledge pieces/chunks. As an expert's power lies in his explanation and reasoning capabilities, the expert system's creditability also depends on the Explanation and Reasoning of the decision made/suggested by the system. Also, human beings have an ability to learn new things and forget the unused knowledge from their minds. Simulation of such learning is essential component of KBS. The life of KBS may vary according to the degree of such simulation. KBS may be either manually updated (manual update) or automatically updated by machine (machine learning). Ideally, the basic frame of a KBS rarely needs to be modified. In addition to all these, there should be an appropriate User Interface, which may have the Natural Language Processing facility. These components are shown in Figure 2.

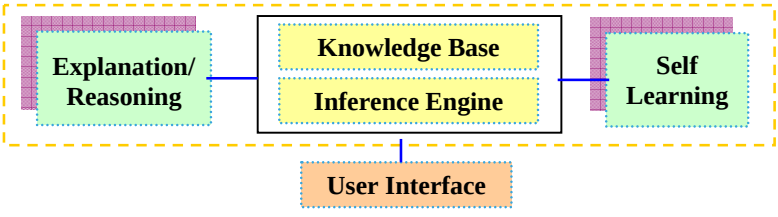


Figure 2: Architecture of a Knowledge-Based System

According to the classification by Tuthhill & Levy (1991), there are main 5 types of the KBS exist:

(i) Expert Systems,

(ii) Hypertext Manipulation Systems,

(iii) CASE Based Systems,

(iv) Database in conjunction with an Intelligent User Interface and

(v) Intelligent Tutoring Systems.

KBS ADVANTAGES AND LIMITATIONS

Knowledge-based systems are more useful in many situations than the traditional computer based information systems. Some major situations include:

- ☞ When expert is not available.
- ☞ When expertise is to be stored for future use or when expertise is to be cloned or multiplied.
- ☞ When intelligent assistance and/or training are required for the decision making for problem solving.
- ☞ When more than one experts' knowledge have to be grouped at one platform.

With the proper utilization of knowledge, the knowledge-based systems increase productivity, document rare knowledge by capturing scare expertise, and enhance problem solving capabilities in most flexible way. Such systems also document knowledge for future use and training. This leads to increased quality in problem solving process. However, the scarcity and nature of knowledge make the KBS development process difficult and complex. The transparent and abstract nature of knowledge is mainly responsible for this. In addition, this field needs more guidelines to accelerate the development process. Following are some of the major limitations with the KBS:

- ☞ Acquisition, representation and manipulation of the large volume of the data/information/ knowledge.
- ☞ High-tech image of the AI field.
- ☞ Abstract nature of the knowledge.
- ☞ Limitations of cognitive science and other scientific methods.

A number of AI/KBS application areas also open up deep philosophical issues. This reveals a promising field of study whose primary concern is to find out more effective ways to understand and apply intelligent problem solving, planning and communication skills to a wide range of practical problems.

KBS DEVELOPMENT

Figure 3 presents the overview of KBS development process. The knowledge of the expert(s) is stored in his mind in a very abstract way. Also every expert may not be familiar with knowledge-based systems terminology and the way to develop an intelligent system. The Knowledge Engineer (KE) is responsible person to acquire, transfer and represent the experts' knowledge in form of computer system. People, Experts, Teachers, Students and Testers are the main users' groups of knowledge-based systems.

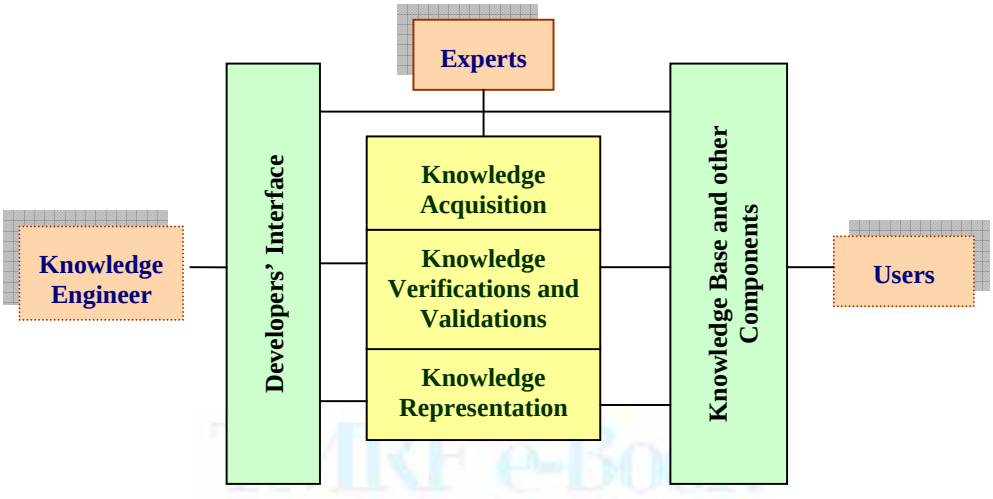


Figure 3: Development of a Knowledge-Based System

The knowledge acquisition process incorporates typical fact finding methods like interviews, questionnaires, record reviews and observation to acquire factual and explicit knowledge. However, these methods are not much effective to extract tacit knowledge which is stored in subconscious mind of experts and reflected in the mental models, insights, values, and actions of the experts. For this, techniques like concept sorting, concept mapping, and protocol analysis are being used.
The acquired knowledge should be immediately documented in a knowledge representation scheme. At this initial stage, the selected knowledge representation strategy might not be permanent. However documented knowledge will lead the knowledge engineer/ developer to better understanding of the system and provides guidelines to proceed further. Rules, frames, scripts and semantic network are the typical examples of knowledge representation scheme. It is responsibility of the knowledge engineer to select appropriate knowledge presentation scheme that is natural, efficient, transparent, and developer friendly. One may think for hybrid knowledge representation strategies like rules within the frames in slots like "on need" and "on request"; semantic network of default frames etc. More about knowledge acquisition, knowledge representation and knowledge-based system development model is available in the book Knowledge-based systems (Akerkar and Sajja 2009).

KBS TOOLS

A KBS tool is a set of software instructions and utilities taken to be a software package designed to assist the development of knowledge-based systems. Personal computers, typical programming languages like java and framework like .NET can also be used in KBS development. These programming languages are general purpose and also being used to develop other application than AI applications. KBS shell with the readymade utilities of self learning, explanation and inference etc. like Java Expert System Shell (JESS), GURU, Vidwan are more specific and can also be useful to develop KBS. Tailor made KBS can be developed using programming languages like LISP and Prolog.

John McCarthy (1960) published a remarkable paper showing a handful of simple operators and a notation for functions, one can build a whole programming language. He called this language Lisp, for "List Processing," because one of his key ideas was to use a simple data structure called a list for both code and data. There are various versions of Lisp available namely KLISP and C Language Integrated Production System (CLIPS).

Prolog is a logic programming general purpose fifth generation (AI) language. It has a purely logical subset, called "pure Prolog", as well as a number of extralogical features. Prolog has its roots in formal logic, and unlike many other programming languages, Prolog is declarative. The program logic is expressed in terms of relations, and execution is triggered by running queries over these relations. The language was first conceived by a group around Alain Colmerauer in Marseille, France, in the early 1970s. According to Robert Kowalski (1988), the first Prolog system was developed in 1972 by Alain Colmerauer and Phillipe Roussel.

Packages software like MATLAB, Java Neural Network Simulator (Java NNS) etc. and markup open sources based tools like Artificial Intelligence Markup Language (AIML) and Project D (developed in AIML and open source) can also be used to develop KBS. Systems which work with multiple agents and intelligent agents may use Knowledge Query Manipulation Language (KQML) for agents' communication. CommonKADS and Protégé also help in assisting KBS development process in user friendly way.

According to Stefan Robertson and John K C Kingston there are approximately 200 KBS tools. Alty (1989) groups the products into three main categories based primarily on functionality which also happen to differ markedly in the platforms on which they are available. These groups are (i) Shells, (ii) Languages, and (iii) Toolkits. Inference ART and KEE were among the first commercially successful toolkits to develop KBS.

Besides support towards knowledge acquisition and representational features, there are other features like price, flexibility, ease of use, user friendliness and vendor availability and support, and documentation support from the tool need top be considered before final selection.

KBS PURE APPLICATIONS

Knowledge-based systems applications are divided into two broad categories namely: (i) pure knowledge-based systems applications and (ii) applied knowledge-based systems application. Pure applications include research contributing in knowledge-based systems and AI development techniques such as knowledge acquisition, knowledge representation, models of automated knowledge-based systems development (knowledge engineering approaches, models and CASE tools for KBS), knowledge discovery and knowledge management types of tools. Table 2 tries to presents a few possible research areas in pure KBS development.

Area	Sub-area	Example Applications
KBS Development	Knowledge acquisition	▪ Automatic knowledge acquisition
	Knowledge representation, Reasoning, explanation and inference	▪ Multi layer KBS ▪ Semantic web ▪ Hybrid representation structures
	Development models, quality standards and protocols	▪ Intelligent software engineering KBS ▪ Automatic programming
Knowledge Management	Knowledge discovery	▪ Knowledge-based data mining ▪ Automatic generation and learning of wrappers ▪ Knowledge creation and reuse
	Knowledge share and disseminations	▪ Knowledge dissemination techniques ▪ Trust based network
Information and Query Based Systems	Information extraction Information retrieval Accessing distributed databases	▪ Meta search engines ▪ Heuristic information filtering functions ▪ Intelligent query formation KBS ▪ Knowledge-based access of data warehouses
User Interface	KBS shell Combining databases with KBS through intelligent user interface KBS and Database Integration	▪ Generic interface in native languages ▪ Customized presentation of information according to user profile ▪ Interface for special target audience say senior citizen or blind people
Expert Systems	Rule based systems Frame based systems	▪ Type 1 and type 2 fuzzy logic based diagnostic systems
Distributed KBS	Network online KBSs Hyper linked KBS Systems on semantic web	▪ Intelligent Routing algorithms
Knowledge Grid	Middleware services Protocols	▪ Deign and implementation of generic k-grid framework
Multi Agent Systems	Agent communication language	▪ Deign and implementation of generic multi-agent framework
Tutoring Systems	e-tutors e-Learning systems	▪ Learning object repositories ▪ General e-content repositories accessed by many applications
Soft computing based KBS	Connectionist system and KBS	▪ Learning paradigms
	Evolving systems	▪ Genetic programming ▪ Genetic fuzzy function ▪ Self evolving NN structures ▪ Automatic generation of rule bases

Table 2: Research Areas in Pure KBS development

KBS FOR INTEGRATED DEVELOPMENT

The Knowledge-based systems can be used for interpretation, prediction, diagnosis, design, planning, monitoring, debugging, repair, instruction, control, etc. Such advanced technology should be made available in urban and rural areas to utilize expert knowledge for holistic development. Such systems export knowledge in underdeveloped and remote area where expertise is rare and costly. Hence, knowledge-based systems KBS should be at the primary consideration while designing the development plan for a nation. The share of AI/KBS systems in IT is improved significantly.
In addition, today's KBS are easier to use, less expensive and integrate well with traditional technologies, so it can provide a fundamental technology to the majority of the applications for today's scenario. The four major dimensions of the rural development process namely; Economical, Social, Physical and Health development are considered for the holistic development. Major resources for development are considered as Natural Resources, Human Resources, Livestock and Agricultural Resources. The KBS applications are classified according to the above dimensions & resources. Figure 4 describes the situation graphically and Table 3 lists some examples in each dimension.

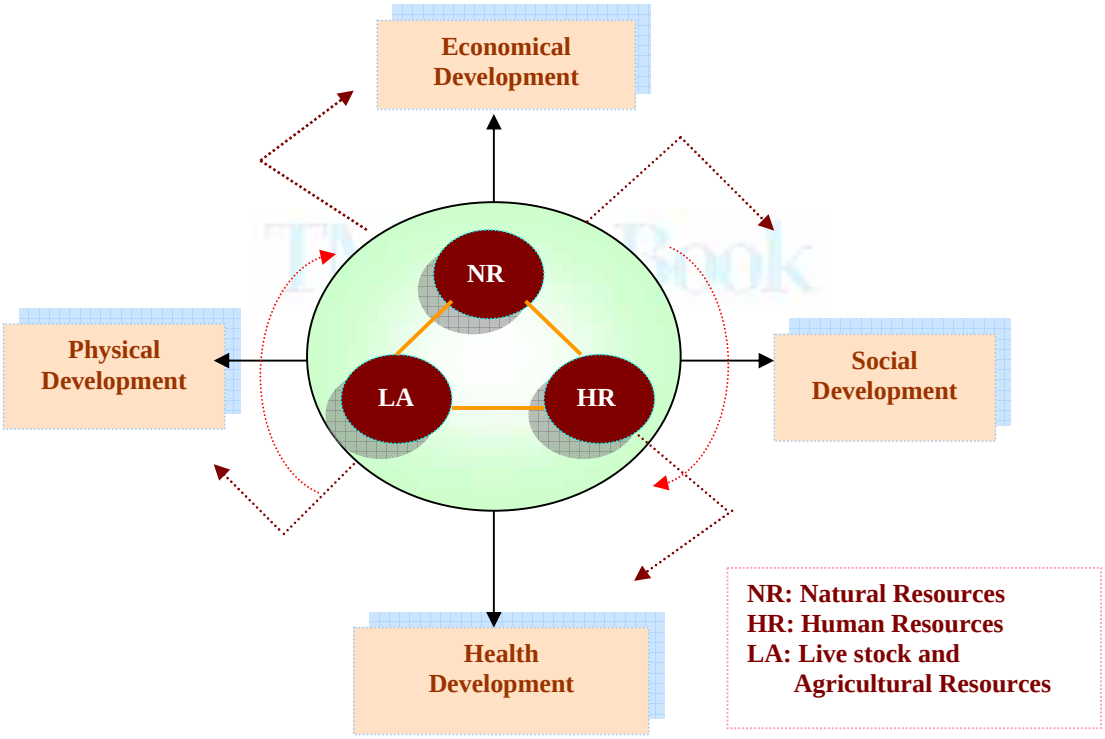


Figure 4: Dimensions of Development through Knowledge-Based Systems

Physical Communication Planning & Administration Forestry, Energy, Agriculture etc.	**Economical** Small Scale Industry Agri-Business & Co-operative etc.
Health Nutrition Sanitation Community Health etc.	**Social** Education & Training Social Awareness Programme etc.

Table 3: Examples of Knowledge-Based Systems in Different Dimensions

Some example of knowledge-based system in the areas listed in the Table 3 can be outlined as follows:

Physical development

KBS for infrastructural planning
Small scale industries like agribusiness and co-operatives
Energy planning and reuse
E-Governance
Irrigation management and supervision
Communication and transportation
Public distribution services
Special programmers like drought prone area planning
Forestry
KBS for natural resource management
Knowledge-based planning for land use and land reform
Network monitoring systems
KBS for resource sand material management
KBS for river and land use management
KBS for soil health card for villagers
KBS for intelligent manufacturing and new product development
KBS for geographic property identification and measuring
Robotics and fly/drive by wire vehicle system
Pilot training and space training through virtual reality
Publication and printing media KBSs
Loan passing system
Fault diagnostic System

Economical Development

Employment exchange services
Market reforms/information systems
New product development advisory
Business selection advisory
Tax planning system
Knowledge-based planning of agricultural products
Knowledge-based planning for agricultural inputs
Knowledge-based diagnosis for plants and animal diseases

Crop land pattern matching system for agriculture
Intelligent manufacturing
Matching buyers and sellers agent in e-commerce
Embedded KBS in the devises like fuzzy washing machine
Robotic and intelligent sensors in manufacturing
KBS for potential risk identification in investments
Software/product quality management
Intelligent ERP based systems

Social Development

Cultural information
Tourism portal
Identity/ration card
Voters identification and election related systems
Intelligent system to identify suitable beneficiaries for Government/NGO schemes
Awareness systems
Child and women health/nutrition systems
Community health
e-learning
Education and training systems
Knowledge-based examination planning system
Games and entertainment KBS
Language translation and tutoring

Health Improvement

Government schemes information system
Diet planning system
Disease diagnostic system
Disease diagnostic system for cattle and live stock
Patient monitoring system
Medical insurance
KBS monitoring in surgical process
KBS for guided neuro-surgery

Knowledge-based systems offer several advantages over humans (Natural Intelligent systems). Some of the major advantages can be listed as follows:

- Knowledge-based systems provide efficient documentation of the important knowledge in a secured and reliable way.
- Knowledge-based systems solve unstructured, large and complex problems in an quick and intelligent fashion and provides justification for the decision suggested.
- Knowledge-based systems offer more than one expert knowledge in an integrated fashion.
- Knowledge-based systems are able to infer (create) new knowledge and learn from cases or data instead of just referring the stored content.

- It is easy to clone and spread knowledge and knowledge-based systems.

Typical information systems deal with data while knowledge-based systems automate expertise and deal with knowledge. Every business in today's competitive world is full of uncertainty and risk. Managing and satisfying customers with quality product/services have become trivial challenge. In this situation the knowledge-based system is wise choice. In spite of plenty of obvious advantages, the knowledge base system development and usage are difficult because of the problems associated (stated earlier in the introductory section) with them. Scientists, researchers and professionals are working on different aspects of knowledge-based system to overcome these limitations and to offer a complete intelligent system which is compatible with the natural intelligent system in controlled fashion.

REFERENCES

Akerkar, R.A. and Sajja, P.S. 2009. Knowledge-based systems: Jones & Bartlett Publishers, Sudbury, MA, USA.

Derek, N. and Jain, H. 1991. The relevance of ES technology for India: A Transborder Perspective, *CSI Digest*, 3(1):68

Pearl, J. 1984. *Heuristic-intelligent search strategies for computer problem solving*: Addison Wesley Publishing Company.

Tuthhill, S. and Levy, S. 1991. *Knowledge-based systems: A managers perspective*: TAB Professional & Reference Books.

McCarthy, J. 1960. Recursive functions of symbolic expressions and their computation by machine. *Communications of the ACM*, 3(4): 184–195.

Kowalski, R.A. 1988. The early years of logic programming, Communications of the ACM, 31(1): 38-43.

Robertson, S. & Kingston, J. 1997. Selecting a KBS tool using a knowledge-based system, AIAI-TR-214. Available at http://www.aiai.ed.ac.uk/project/ftp/documents/1997/97-paces97-select-kbs-tool.ps

Alty J.L. 1989. Expert system building tools. In: eds. Guida, G. and Tasso, C. Topics in Expert System Design: Methodologies and Tools: North-Holland, Amsterdam.

TMRF e-Book
Advanced Knowledge Based Systems:
Model, Applications & Research
(Eds. Sajja & Akerkar), Vol. 1, pp 12 – 28, 2010

Chapter 2

Representing Knowledge Effectively Using Indian logic

Mahalakshmi G.S. and Geetha T.V. *

Abstract: Knowledge becomes the key factor for intelligence. Representing knowledge is the fundamental requirement for Inference and reasoning mechanisms. Inferences will prove efficient only when knowledge is represented and retrieved more naturally. In other words, the pattern of knowledge representation should match human way of knowledge representation to enable mimicking of human inferences. Every research in artificial intelligence had proposed striking advances in knowledge representation mechanisms. This paper discusses such an aspect of knowledge representation adapted from Indian philosophy. The paper also presents a short comparison with other knowledge representation techniques.

Keywords: Logic, Knowledge representation, Indian philosophy, Nyaya Sastra, Ontology

1. INTRODUCTION

One of the fundamental issues in artificial intelligence is the problem of knowledge representation. Intelligent machines must be provided with a precise definition of the knowledge that they possess, in a manner, which is independent of procedural considerations, context-free, and easy to manipulate, exchange and reason about. Any comprehensive approach to knowledge representation has to take into account the inherently dynamic nature of knowledge. As new information is acquired, new pieces of knowledge need to be dynamically added to or removed from the knowledge base.

For inferences, decision-making, dialogue-exchange or machine learning, the fundamental issue involved is the utilisation of reasoning. Reasoning is the process of arriving at new conclusions. To reach a conclusion, we generally conclude certain investigations. Therefore, if the investigations are not formally represented using a knowledge representation language which is clear and user-friendly, performing reasoning shall become a daunting task.

Lau et al (2003) have outlined a research agenda for automatically acquiring procedural knowledge for use in autonomic systems. It is based on learning procedures by observing experts perform these procedures on live systems, and dynamically building a procedural model that can be executed on a new system to repeat the same task. Over time, as more traces are collected, the procedural model is

* Department of Computer Science and Engineering, Anna University, Chennai 600025, India
mahalakshmi@cs.annauniv.edu, tvgeedir@cs.annauniv.edu

updated. This dynamic learning process also enables the procedure to adapt to changing conditions over time.

A Learnable behavioral model for autonomous virtual agents (Toni and Daniel 2006) and a knowledge delivery service has also been proposed. At this juncture, social learning in robots is worth-mentioning. A reinforcement learning model which deals with human teachers teaching robots has been proposed (Thomaz et al 2006). Here, the learner robots contribute to learning by revealing their internal states to help guide the teaching process. Teacher and learner read and respond to each other, to more effectively guide the learner's exploration.

Virtual communities are relatively a new approach to distributed knowledge sharing. Research in virtual communities show an increase in the need to share knowledge across a distributed environment in present web-based lifestyle. Setting-up successful virtual communities represents an effective way for facilitating the circulation of knowledge in organizations and groups. Perceptual, deliberative, operational, and meta-cognitive means of attention management for virtual community members have also been attempted. Use of Ontology facilitates the sharing of world knowledge between virtual communities (Davies et al 2004). Dynamic matching of distributed ontologies facilitates co-operative learning in virtual knowledge communities.

In a nutshell, with the rise of intelligent front ends to AI systems, it has become conventional to represent the inputs in natural languages and thereafter, the world knowledge embedded in the input is extracted with the help of common sense ontologies. The reason is to reduce the information overload to AI systems by avoiding extensive pre-processing. However, introduction of ontologies to AI systems have a great impact on the performance and accuracy of operation of most systems. Practical application of Ontology based systems in real-time was quite challenging in spite of the benefits of Ontology to modern AI society.

2 ONTOLOGY AND KNOWLEDGE REPRESENTATION

In the context of multiple entities participating in knowledge sharing, a common Ontology can serve as a knowledge-level specification of the ontological commitments of a set of participating entities that are involved in discussion. Ontology is a formal mechanism of representing the world knowledge, out of which effective and easy reasoning is possible during knowledge sharing. Ontological commitments are agreements that define a clear boundary of how to *view* the Ontology. This means that the specification format is independent of the internal representation such that the ontological commitments are defined only at the knowledge level.

Inconsistencies are conflicting information present within the ontologies which inhibit reasoning and inference procedures. Inferences made with such approaches will never be adequate enough to better reasoning while knowledge sharing. The most challenging and important problem is *Ontology evolution*, which is the problem of modifying an Ontology in response to a certain change in the domain or its conceptualization. As Flouris et al (2006) summarises, there are several cases where Ontology evolution is applicable: change in the world knowledge, change in the domain, change in the specification or conceptualisation, change in the perspective, change in user needs or change in belief. This is the strong reason behind our motivation for identifying improvements in knowledge representation.

3 KNOWLEDGE REPRESENTATIONS – PROPERTIES AND CHALLENGES

3.1 Properties of knowledge representation

Knowledge representation cannot be defined in pure epistemological terms. Representation and reasoning are intertwined with knowledge representation. The attempt to deal with representation as knowledge content alone leads to an incomplete conception where reasoning may be put aside. The use of a representation as a medium of expression and communication matters because we must be able to speak the language in order to use it. If we cannot determine how to say what we are thinking we cannot use the representation to communicate with the reasoning system. Several measures of a good knowledge representation may be listed as follows:

- Support to efficient reasoning
- Expressivity – how expressive the knowledge is
- Adequacy – is the represented knowledge adequate
- Satisfiability – role of knowledge which satisfies the goal
- Quality – quality of knowledge within the knowledge representation
- Uncertainty – how much certain the expressed knowledge is
- Consistency – how much consistent the knowledge is

3.2 Challenges of Knowledge representation

The principal notion of a good knowledge representation should be to understand and describe the richness of the world. Apart from this, other challenges are as follows:

- to cope with dynamism in the world knowledge
- to preserve consistency of information across domains
- to accept belief revisions in the knowledge covered under representation
- to represent beliefs in a easy and resourceful manner
- to facilitate better reasoning and inferences over the represented knowledge
- to enable changes from the knowledge definition perspective
- to adapt to addition of necessary information with change in the specification or conceptualization.

The above notion can be better understood by describing more abstractly, the five roles of knowledge representation (Davis et al 1993):

- A knowledge representation is most fundamentally a surrogate; used to enable an entity to determine consequences by thinking rather than acting.
- It is a set of ontological commitments
- It is a fragmentary theory of intelligent reasoning expressed in terms of three components; (1) the representation is fundamental conception of intelligent reasoning (2) the set of inferences the representation sanctions and (3) the set of inferences it recommends.
- It is a medium for pragmatically efficient computation
- It is a medium of human expression; i.e. a language in which we say things about the world

4 VARIETIES OF KNOWLEDGE REPRESENTATION

In the field of AI, many researchers have addressed the problem of knowledge representation; in this area semantic networks played an important role. In semantic networks the knowledge is described by nodes and links. While Quillian aimed at the representation of word meanings [Quillian 68], semantic networks have also been used to model propositions, events, spatial relationships and so on. Since semantic networks failed in providing a unique semantic interpretation, several researchers examined the "semantics of semantic networks" (Woods 75, Brachman 79).

4.1 Production Systems

Production systems (Newell, 1973) are a form of knowledge representation which provides for a good deal of flexibility. Productions are condition-action pairs. The set of productions by itself is quite unstructured. To make it work it needs two kinds of control processes. One is a short-term memory buffer: only the data currently held in that buffer can activate the condition of a production; thus the flow of data in and out of short-term memory determines in part what productions are executed. However, since it will frequently be the case that more than one production condition matches the data in the short-term memory buffer, some kind of conflict resolution procedure is required. Production systems have their disadvantages, too. It is not easy to understand what actually happens in a large production system. The reason is that the interactions within productions may have surprising outcomes.

4.2 Associative networks

An associative network is a structure in which the nodes are unanalyzed concepts, and the links are unlabeled, but vary in strength. Knowledge is thus represented as a network of ideas, with interconnections determined by the laws of association like resemblance, contiguity, and cause-effect, to name a few. While associative nets are messy, perceptually based, semantic nets are orderly and conceptually based. There are different categories of semantic nets [Brachman 1979]. They are:

1. Logical nets: links represent logical relations and nodes are predicates and propositions
2. Epistemological nets: Links are inheritance and structuring relations and nodes are concept types, rather than particular concepts
3. Conceptual nets: Links are semantic relations; nodes are primitive objects or actions
4. Linguistic nets: Primitives are language-dependent; meanings are derived from context and it changes as the network grows

4.3 Frames

The major leap forward from semantic nets was towards more structured knowledge representation, called frames [Minsky 75]. A frame is a data structure that is typically used to represent a single object, or a class of related objects, or a general concept (or predicate). Frames are typically arranged in a taxonomic hierarchy in which each frame is linked to one parent frame. A collection of frames in one or more inheritance hierarchies is a knowledge base. Frames consist of a heading and various slots. Each slot has its default value which can be activated if no other information is available. Thus, as soon as a frame is invoked, a great deal of well-organized knowledge becomes available, without the need for elaborate computations.

The slots of a frame describe attributes or properties of the thing represented by that frame. In addition to storing values, slots also contain restrictions on their allowable values. Every slot has several

components like slot name, value, value restriction, procedure, justification etc. which are referred to as facets. Frames typically allow the specification of default slot values, perspectives and attached procedures. Collections of frames can be combined to frame-systems. The expressive power of frame systems makes it impossible to provide a well defined semantics for them. Both, elements of different network formalisms and basics of the frame theory, have influenced the KL-ONE like representations.

4.4 Conceptual graphs

The basic building blocks of KL-ONE representations are "concepts", i.e. structured conceptual objects. "Roles" are possible relationships between two concepts. The subsumption relation organizes the concepts in concept taxonomy. Concepts are described with respect to their super concepts by restricting and differentiating roles. In particular, roles can be restricted by the number (number restriction) and the range (value restriction) of allowed role fillers. If the specified restrictions constitute necessary and sufficient conditions for the concept, it is called a defined concept, whereas primitive concepts only need necessary conditions. Classification, an important inference mechanism of KL-ONE like systems, inserts concepts at the correct place in the concept hierarchy.
A logical reconstruction of KL-ONE revealed that the semantic status of a number of notions of KL-ONE was rather unclear. TSLs are formal knowledge representation languages derived from KL-ONE providing well-defined semantics which enables the decision whether the inferences are sound and complete. A number of KR systems based on TSLs have been developed, for instance, Krypton [Brachman et al. 85], KL-Two [Vilain 85], Back [Peltason et al. 89], Loom [MacGregorBates 87]. Besides a component for defining concepts and reasoning about the relationships between concepts (terminological component, TBox) these systems include an assertional component (ABox) that allows the definition of assertions about individuals.
A conceptual graph is a network of concept nodes and relation nodes (Conceptual Graph Standard 2002). The concept nodes represent entities, attributes, or events (actions) while the relation nodes identify the kind of relationship between two concept nodes. The main characteristics of conceptual graphs are:

- Concepts and relations replace predicates and arguments from predicate logic
- A relation's signature defines what concept types it can relate
- Concepts allow referents to specify an individual or a set of individuals of a certain type
- A type hierarchy can be defined on concepts
- Different graphs can be related through a co reference link on an identical concept

4.5 Description Logics

Description logics (DL) are both class-based and logic-based knowledge representation languages, which allow for modeling an application domain in terms of objects, classes and relationships between classes, and for reasoning about them (Baader et al 2002). Unlike object-oriented languages used in databases and programming languages, DL permit the specification of a domain by providing the definition of classes, and by describing classes using a rich set of logical operators. By using DL, one can specify not only the necessary conditions that objects of a given class must obey, but also the sufficient conditions for an object to belong to a certain class.
A knowledge base built using DL is formed by two components: the intensional one, called Tbox, and the extensional one, called ABox. The basic building blocks are concepts, roles and individuals. Concepts describe the common properties of a collection of individuals and can be considered as unary predicates, which are interpreted as sets of objects. Roles are interpreted as binary relations between objects. Each DL defines also a number of language constructs (such as intersection, union, role

quantification, etc.) that can be used to define new concepts and roles. DL support inference patterns which occur in many applications of intelligent information processing systems.
The techniques for reasoning in DL, refers to four different settings: 1) reasoning with plain concept expressions 2) reasoning with instances of concepts 3) reasoning with axioms expressing properties of concepts 4) reasoning with both instances of concepts and axioms. One of the main reasoning services of a DL system is to automatically build the subsumption hierarchy of classes, i.e., a graph showing all the subsumption relations between the classes of an application. The classification hierarchy of Ontology expressed as per DL provides useful information on the connection between different concepts, and it can be used to speed-up other inference services. The following section gives an insight into quite a few practically implemented DL systems and knowledge sharing systems.
KL-ONE, CLASSIC, BACK, LOOM, KRIS, CRACK are knowledge representation systems built on DL. KL-ONE (Brachman and Schmolze 1985) is one of the typical systems of earlier generation DL systems. KL-ONE inherently included the notion of inferring implicit knowledge from given declarations. It also supported generic concepts, which denote classes of individuals, and individual concepts to denote individuals. More important and unique to KL-ONE is the core idea of providing ways to specify concept definitions allowing a knowledge engineer to declare the relations between high-level concepts and lower-level primitives.
CLASSIC is a knowledge representation system (Alex et al 1989) based on DL designed for applications where only limited expressive power is necessary, but rapid responses to questions are essential. LOOM introduced the notion of Tbox and Abox for dealing with concept definitions and assertion regarding the concepts respectively (MacGregor 1991). The LOOM was developed for applications like natural language and image interpretation. A problem with the LOOM approach is that it is hard to characterize the source of incompleteness of reasoning algorithms, which might lead to unexpected behavior.
BACK is based on DL, implemented in PROLOG. The BACK architecture was designed to support incremental additions to the Abox and retraction of old information. Abox assertions can be retrieved from a database by automatically computing SQL queries. The BACK system was considered highly suitable for the applications where reasoning about time was important. FLEX (Quantz et al 1996), a successor of BACK included the facility for reasoning about equations and inequalities concerning integers. The CRACK system (Bresciani et al 1995) supported the DL language with all basic extensions in the constructor zone and the inference algorithms. KRIL (Aghila et al 2003) is based on extended DL which takes inspirations from Indian philosophy. KRIL basically provided all fundamental definitions and reasoning services of every other DL systems but the definition of knowledge base in KRIL followed the systems of Indian Logic.

4.6 Extended Description Logics

DL may be suitable for effective knowledge representation from a detailed knowledge base. However, DL systems failed to construct the atomic concept in terms of its member qualities. Also the relations defined possessed heavy limitations with respect to their scope and were only superficial, i.e. between atomic concepts. Extended DL system, besides satisfying the relation definitions at the conceptual level, also associated relations between concepts and its member qualities. Thus, by enhancing the number and type of relations that can be defined at the highest level of abstraction, inference becomes multi-relational rather than been restricted to the normal hierarchical, part-of and instantiation relations.

4.8 The OAR Model

The object-attribute relational model is one worth mentioning at this juncture. The nodes in OAR model are objects; links are relations. The objects are associated with attributes of their own. The novelty in OAR model (Wang 2006a) is that it matches the process of human cognition.
Indian philosophy serves as the strong foundation for the mathematical support behind the construction of knowledge base which is in the form of Indian logic based ontologies. In this context, Nyaya Sastra, the famous Indian school of Philosophy, has laid great stress on categorisation of world knowledge into a classification framework (Virupakshananda 1994). The following section elaborates on the motivation behind the selection of Indian philosophy for the new knowledge representation formalism discussed in this paper.

5 MOTIVATION

For representing Ontology in a more meaningful manner, description logics (DL) have been used widely (Calvanese et al 2002). However, though the representation formalisms like DL, has the capability to deal with uncertain knowledge, they do not consider the presence of member qualities as an 'inherent' part of concept descriptions. In addition, relations defined possessed heavy limitations with respect to their scope and only existed between atomic concepts. Due to the added requirements of Ontology representation based on Nyaya, extensions were made both at the concept-constructor zone and the relation zone (Mahalakshmi et. al, 2002).
Modification of basic reasoning services of standard DL by Aghila et al (2003) created scope for definition of more descriptive concept elements based on Indian philosophy. However, extended DL Systems (Mahalakshmi et. al, 2007, Mahalakshmi and Geetha, 2008a) dealt with the basic reasoning services from philosophical perspective but the ontological classification was present only as part of reasoning mechanisms. Though concepts possessed 'qualities' defined by Nyaya Sastra, general relations existed between two concepts, and a special relation, 'inherence' existed (for applicable concepts) between concept and its member quality. i.e. the definition of ontological entities provided by extended DL does not tackle the special requirements dictated by Nyaya Sastra's classification framework.
Other Nyaya recommendations like 'invariable concomitance' relation, 'contact-contact' relation and the like (Virupakshananda 1994) were not incorporated in extended DL Systems (Aghila et al 2003). This motivated us to design a special knowledge representation formalism which completely inherits all the classification recommendations of Nyaya Sastra.

6. NORM – NYAYA ONTOLOGY REFERENCE MODEL

Nyaya Ontology Reference Model (NORM) defines the standards for constructing Ontology, based on the recommendations of the epistemology definitions of Nyaya-Vaisheshika schools of Indian philosophy. NORM (Mahalakshmi G.S. and Geetha T.V. 2008b), is organized as two-layer Ontology, where the upper layer represents the abstract fundamental knowledge and the lower layer represents the domain knowledge. According to NORM, a node in the Ontology is composed of an enriched concept which is related implicitly to its member qualities and explicitly to other peer concepts, by means of relations (Wada 1990).
A node of Nyaya-Vaisheshika based Ontology has the following structure (Figure 1). Every concept of the world knowledge shall be thoroughly classified as per NORM structure. The abstract and domain concepts form a strict classification hierarchy. The traits of the top-level concepts are applicable down

the hierarchy. Every concept in the NORM model has links to other concepts by external relations (Figure 1a). A concept is made of qualities. In addition, the qualities are bounded to the concept by internal relations. The qualities may also be related to each other, which are depicted as dotted edges (Figure 1b). Every quality has a set of values, and the values are bounded to the qualities by grouping relations (Figure 1c). This model is inspired by the various recommendations of classifications of world knowledge according to Nyaya-Vaisheshika.

According to NORM, a concept is an abstract entity which embeds several qualifying attributes of its own. The attributes are bound to the concept by relation existence. An attribute is a sub-property of a concept / relation which is used to define the concept set. Attributes are optionally supported by values. Quality may be 'mandatory' or 'optional' type. The 'mandatory' attribute of a concept shall be made 'exclusive' or 'exceptional' but not 'optional'. The 'optional' attribute of a concept shall be made 'mandatory' which contributes more to inferential learning; but the vice versa is not allowed and is considered as violation of Ontology definitions. The qualifying attributes of a concept are not present as only the feature set of that concept; instead, every qualifying attribute is related to or bound to the concept by relations. Relations may be invariable, exclusive, exceptional etc. The following section discusses the system of classification recommended by Nyaya Sastra.

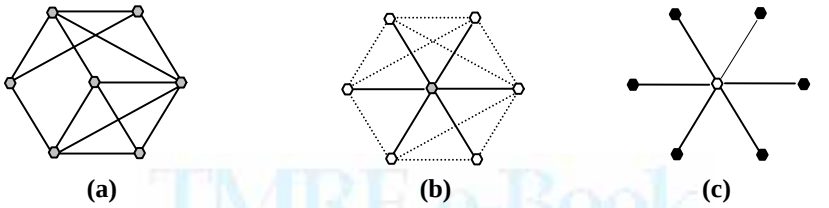


Figure 1 NORM Model for Cognitive Knowledge representation

(a) Ontology with concepts as nodes and external relations as edges, (b) a concept with qualities as nodes, internal relations as thin edges, tangential relations as dotted edges, (c) a quality with values as nodes, grouping relations as edges (Mahalakshmi G.S. and Geetha T.V. (2008b),)

7 Nyaya System of Classification

According to Nyāya Sastra (Sinha and Vidyabhusana 1930), every concept is classified into seven categories: substance, quality, action, generality, particularity, inherence and negation. Among these, the substance is of nine kinds: earth, water, light, air, ether, time, space, soul and mind. Every substance is threefold: body, organ and object. The object of light is fourfold: earthly, heavenly, gastric and mineral. Every substance is said to possess some quality. The quality is of twenty-four varieties which in turn possess values (Figure 2).

The permissible action associated with the substance is of five types: upward motion, downward motion, contraction, expansion, and motion. Generality is either more comprehensive or less comprehensive. Negation is of four varieties: antecedent negation (or prior negation), destructive negation (or posterior negation), absolute negation and mutual negation. Out of the nine substances, odour persists only in earth and is inherent. Earth exists in all the seven colors. Air has no color; water is pale-white in color and light is bright-white in color. Air has touch. Water has cold-touch and light has hot-touch. Dimension (or magnitude), distinctness, conjunction and disjunction are present in all the nine substances. Remoteness and Proximity is found in earth, water, light, air and mind. Heaviness or Weight is only in earth and water. Viscidity is present only in the substance, Water.

Color: white, blue, yellow, red, green, brown, varied
Taste: sweet, acid, saline, pungent, astringent, bitter
Odour: fragrant, foul
Touch: cool, hot, lukewarm
Number
Magnitude: atomic, large, long, short
Separateness
Conjunction
Disjunction
Remoteness: spatial, temporal
Proximity: spatial, temporal
Weight
Fluidity: natural, artificial
Viscidity
Sound: articulate, inarticulate
Intellect
Pleasure: ***Remembrance, apprehension***: *Erroneous apprehension, valid apprehension*:
Valid apprehension : perception, inference, analytical knowledge and verbal testimony
Pain
Desire
Aversion
Volition
Merit
Demerit
Tendency

Figure 2 Ontological Classification of Nyāya-Vaiśesika Qualities

8 NYAYA DESCRIPTION LOGIC (NDL)

The knowledge representation language of 'Gautama' is christened as 'Nyaya Description Logics' (NDL). The NDL commands are classified into concept/relation definition language (CRDL), concept/relation manipulation language (CRML) and a set of editing commands and a query language (Mahalakshmi and Geetha 2008a). This knowledge representation language can be further used to define, manipulate and query the various levels of knowledge. CN refers to Concept name, QN refers to Quality Name, V – Quality value (Ex: color – Indigo: quality: color, value: Indigo) RN refers to Role name, I refer to Instance and Rdesc refers to Role descriptions. Following the above norms of definition of knowledge representation languages (as description logic commands), here, we define the sample Nyaya logic commands which are listed in Table A1.1.
The CRDL and CRML commands are seldom needed for creating the Ontology since, 'Gautama', the Ontology editor provides enough facilities for creation and updation services. The query language shall be used with the RDF generated by Gautama, to query about various parts of the Ontology. Here, we discuss few commands of the query services.

- Concept-satisfiable – This takes a concept name as the parameter and checks whether the addition of the concept will not violate the Ontology definitions that exist prior to the execution of this command
- Concept-subsumes – This takes two concepts as input, and checks whether the first concept subsumes the second concept. This is one of the famous reasoning service provided by any Ontology-based reasoner.
- Concept ancestors and Concept-descendants – These commands list the ancestral / descending concepts in the Ontology hierarchy. Role-ancestors and Role-descendants also have similar purpose.
- Sub-concept, Super-concept – These commands retrieve the child nodes or parent nodes of the parametric concept from the Ontology hierarchy
- Chk-concept-related – This command has three variations. It either checks whether a concept is related to another concept, through a particular relation name or through a particular set of relation categories.
- Chk-quality – This command checks the entire Ontology hierarchy to check if the required quality is available in the Ontology
- Chk-concept-quality – This command checks the entire Ontology hierarchy to check if the particular concept has the required quality.
- All-concepts, all-qualities, all-roles, all-instances – These commands just lists all the concepts, qualities, roles or instances available in the Ontology.
- Retrieve-direct-concepts, retrieve-indirect-concepts – The first commands take an instance as input, and retrieve all the directly related concepts to those instances; The second command take the instance as input and retrieves all the second and higher degree concepts related to those instances. For example, if 'TooToo' is an instance of penguin, the first command may retrieve 'penguin' as the result; the second command will retrieve all the ancestors of penguin which are conceptually related to penguin. Retrieve-direct-instances, retrieve-indirect-instances also serve the same purpose.

The following are the various tags defined for the RDF of Indian logic Ontology generated according to NORM.

- <rdf:concept> - This tag is used to declare a concept prior and after its definition
- <rdf:name> - This tag is used to declare the name of a concept / quality / relation.
- <rdf:desc> - This tag is used to create descriptions or definitions for a particular concept
- <rdf:axiom> - This tag is used to create concept axioms
- <rdf:quality> - This tag is used to create member qualities for a given concept
- <rdf:type> - This tag is used to declare the type of a concept / quality / relation
- <rdf:role> - This tag is used to declare the role of a concept / quality
- <rdf:category> - This tag is used to declare the category of relation like external, internal, tangential or grouping.
- <rdf:operator> - This tag is used to declare the logical operators like and, or while creating the concept axioms of the Ontology

'Gautama' is a tool (Mahalakshmi and Geetha, 2009a) which facilitates the building of Ontology according to Nyaya Sastra's classification scheme. It should equally be possible to build the Ontology

from the scratch, without using the tool and just by opening a notepad and writing the concepts along with the defined RDF tags for Indian logic ontologies.
The sample RDF generated for a simple Ontology for 'birds' domain is given in Figure 3. The facilities for interacting with the knowledgebase are done through knowledge representation languages. These languages form part of the reasoning mechanisms of RDF schema.

```
<rdf:concept>
        <rdf:name>bird</rdf:name>
</rdf:concept>

<rdf:concept>
        <rdf:name>pigeon</rdf:name>
        <rdf:axiom>bird</rdf:axiom>
        <rdf:desc>
                <rdf:quality>
                        <rdf:name>action</rdf:name>
                        <rdf:type>exclusive</rdf:type>
                        <rdf:operator>and</rdf:operator>
                        <rdf:value>
                                <rdf:name>walk</rdf:name>
                                <rdf:operator>and</rdf:operator>
                        </rdf:value>
                        <rdf:value>
                                <rdf:name>fly</rdf:name>
                                <rdf:operator>inclusive</rdf:operator>
                                <rdf:role>
                                        <rdf:name>isa</rdf:name>
                                        <rdf:category>VVrelationship</rdf:category>
                                        <rdf:type>symmetric</rdf:type>
                                        <rdf:relToValue>walk</rdf:relToValue>
                                        <rdf:operator>and</rdf:operator>
                                </rdf:role>
                        </rdf:value>
                        <rdf:role>
                                <rdf:name>hasA</rdf:name>
                                <rdf:category>QVrelationship</rdf:category>
                                <rdf:type>transitive</rdf:type>
                                <rdf:relToValue>walk</rdf:relToValue>
                                <rdf:operator>or</rdf:operator>
                        </rdf:role>
                        <rdf:role>
                                <rdf:name>hasA</rdf:name>
                                <rdf:category>QVrelationship</rdf:category>
                                <rdf:type>transitive</rdf:type>
                                <rdf:relToValue>fly</rdf:relToValue>
                                <rdf:operator>and</rdf:operator>
                        </rdf:role>
                        <rdf:role>
                                <rdf:name>isA</rdf:name>
                                <rdf:category>QCrelationship</rdf:category>
                                <rdf:type>transitive</rdf:type>
                                <rdf:relToConcept>pigeon</rdf:relToConcept>
```

```
                        <rdf:operator>and</rdf:operator>
                    </rdf:role>
                </rdf:quality>
                <rdf:quality>
                    <rdf:name>consumes</rdf:name>
                    <rdf:operator>and</rdf:operator>
                    <rdf:type>exclusive</rdf:type>
                    <rdf:role>
                        <rdf:name>isA</rdf:name>
                        <rdf:category>QQrelationship</rdf:category>
                        <rdf:type>transitive</rdf:type>
                        <rdf:relToQuality>action</rdf:relToQuality>
                        <rdf:operator>and</rdf:operator>
                    </rdf:role>
                    <rdf:role>
                        <rdf:name>isA</rdf:name>
                        <rdf:category>QCrelationship</rdf:category>
                        <rdf:type>transitive</rdf:type>
                        <rdf:relToConcept>pigeon</rdf:relToConcept>
                        <rdf:operator>and</rdf:operator>
                    </rdf:role>
                </rdf:quality>
                <rdf:role>
                    <rdf:name>isA</rdf:name>
                    <rdf:category>CCrelationship</rdf:category>
                    <rdf:type>transitive</rdf:type>
                    <rdf:relToConcept>bird</rdf:relToConcept>
                    <rdf:operator>and</rdf:operator>
                </rdf:role>
            </rdf:desc>
        </rdf:concept>
```

Figure 3 NORM RDF to describe the concept 'pigeon'

Table 1 NDL commands for querying with Gautama

CRDL	**CRML**
define-concept<*CN, Level*>	insert-quality<*QN*>
define-concept-axiom<*CN,Cdesc*>	delete-quality<*QN*>
disjoint-concept<*C1,C2*>	insert-values<*QN,V1….Vn*>
define-role-axiom<*RN, Rdesc*>	delete-values<*QN,V1….Vn*>
disjoint-role<*R1,R2*>	delete-concept<*CN*>
define-concept-role<*RN,C1,C2*>	delete-instance<*I*>
define-concept-qualities<*CN, (QM,Qman.List) / (QO,Qopt.List) / (QE,Qexceptional.List) / (QX,Qexclusive.List)*>	update-instance<*I,Cnold,Cnnew*>
define-quality-values<*CN,QN,V1….Vn*>	delete-role-filler<*I1,I2,RN*>
define-role-quality <*RN,CN, Qreflexive.List / Qsymmetric.List / Qassymmetric.List / Qantisymmetric.List / Qtransitive.List / Qdirect.List / Qindirect.List / Qexclusive.List*>	update-role-filler<*I1,I2,Rnold,Rnnew*>
	delete-role<*RN*>
	insert-role<*RN*>
	delete-concept-quality<*CN,QN*>
	delete-quality-value<*CN,QN,VInvariableConcommitance.List / VExclusive.List / VInvariableConcommitance.List*

define-quality-role<*RNreflexive.List / RNasymmetric.List / RNsymmetric.List / RNantisymmetric.List / RNtransitive.List / RNdirect.List / RNindirect.List / RNexclusive.List, CN,QN*>	*/ VDirect.List* > insert-quality-value<*CN,QN,VInvariableConcommitance .List / VExclusive.List / VInvariableConcommitance.List / VDirect.List*> update-quality-value<*CN,QN,Vold,Vnew*>

Table 1 (Continued)

Query language	Query language
concept-satisfiable<*CN*>	retrieve-direct-concepts<*I*>
concept-subsumes<*C1,C2*>	retrieve-indirect-concepts<*I*>
concept-disjoint<*C1,C2*>	retrieve-concept-fillers<*RN,C1*>
chk-concept<*CN*>	all-concepts<*I*>
concept-atomic<*CN*>	retrieve-qualities<*CN*>
concept-ancestors<*CN*>	retrieve-quality-value<*CN,QInvariableConcommitance / QExclusive / QExceptional*>
concept-descendants<*CN*>	retrieve-quality-value<*CN,QDirect*>
super-concept<*CN*>	chk-instance<*I*>
sub-concept<*CN*>	chk-instance-type<*I,CN*>
chk-concept-related<*C1,C2*>	chk-instance-related<*I1,I2*>
chk-concept-related<*C1,C2,RN*>	retrieve-direct-instances<*I*>
chk-concept-related<*C1,C2,RNreflexive.List / RNasymmetric.List / RNsymmetric.List / RNantisymmetric.List / RNtransitive.List / RNdirect.List, RNindirect.List / RNexclusive.List* >	retrieve-indirect-instances<*I*>
	retrieve-instance-fillers<*RN,I1*>
	all-instances<*CN*>
	retrieve-related-instances<*RN*>
chk-quality<*QN*>	retrieve-quality-value<*I,QN*>
chk-concept-quality<*CN,QN*>	chk-role<*RN*>
all-qualities	all-roles
	role-descendants<*CN*>
	role-ancestors<*CN*>

The Ontology editor based on Indian logic facilitates the user to carefully handcraft the Ontology based on Nyāya-Vaiśesika system of classification of concepts in the required domain. However, there are other noteworthy projects existing in the knowledge representation arena. Cyc, WordNet, Concept-Net and Mind-Net are to name a few (Matuszek et al 2005; Fellbaum 1998; Vanderwende et al 2005; Concept Net 2008). In future, adapting more ideas of building the Ontology from Indian philosophy would strengthen the outcome of the Ontology editor.
Thus, with a detailed manner of representing world knowledge, we could arrive at better inferences during common-sense reasoning, which will serve as a major leap forward in knowledge representation and interpretation services.

9 CASE STUDY

This section shortly discusses the case study across three knowledge representation formalisms namely, description logics, extended description logics and nyaya description logics.

Let us consider the example: *Cat is a mammal*

In DL, there will be two concept nodes *cat* and *mammal* generated and a relation *is-a* links between two concept nodes. In X-DL (extended DL) there will be same two concept nodes created and relation applied, but the new nodes will have more details added to it, namely, *cat is a living thing*. The reason is the class *mammal* has quality *soul* associated to it (Virupakshananda 1994), thus defining *cat* to be a *living thing*. At this juncture, Nyaya description logics also does the same. This is the same for *Penguin is a bird*. When a new statement like, *Will penguin fly?* comes, DL reasoners fail to say the correct answer. In X-DL, the attributes of *penguin* will be elaborated as '*one with boneless wings*' in the axiom of *penguin*, therefore, X-DL and NDL would be able to justify that *penguin will not fly*.
However, for *mountain is smoky*, DL reasoners tend to do a little over the inferences. X-DL reasoners analyses the attributes of both the concepts *mountain* and *smoke* and tries to relate both the concepts by other intelligent means of relations like *smoke is over the mountain* and *mountain is not made of smoke*. But only NDL reasoners tend to analyse the reason of *smoke* over *mountain* and give out their inferences as *mountain has fire*. They associate the invariable concomitance relation (Virupakshananda 1994) between *smoke* and *fire*. Invariable relation says that 'wherever x is present y is also present'. Thus wherever smoke exists, fire also exists. NDL reasoners mimick the way of human inferences (Mahalakshmi and Geetha 2009b) by finding the cause of the effect smoke over the mountain.
The reason is obvious. As said earlier, NDL does not only vary in defining the concept axioms but rather the classification of world entities take inspirations from Nyaya Ontological classification, and therefore, a concept or an instance of a concept is properly tagged and classified under its definitional hierarchy.
At this juncture, a small comparison with OAR model (Wang 2006b) would be wonderful. OAR model, also called as Object-Attribute-Relational model defines the concept axioms though in the style of human cognition, but not adequately enriched in relation with other existing information in the Ontology. For example, *cat is a mammal* would require only two object nodes *cat* and *mammal* with a link *is-a* between them, as in the case of DL reasoners. However, the improvement in OAR model is that the properties of object nodes are also taken into consideration while defining the object nodes. In NDL, we refer to the object nodes of OAR model as to concept nodes and instance nodes, and therefore, the definition of *cat* as a concept axiom would be more detailed as to defining its properties, relations between its properties, permissible values of the properties etc.
In other words, *an yellow cat* will be defined as a *cat* object with *yellow* as the object property in OAR model; in NDL it is like defining a *cat* concept and *yellow* as quality: color, to be inserted into the knowledge base. However, the intelligence lies in the inference algorithm that, by commonsense definitions into NDL, a cat can never be yellow in color, and therefore, the *yellow* color may be due to some artificial coloring and therefore, this intelligence helps in eliminating inconsistencies from the knowledgebase.

10 CONCLUSION

This paper discussed the Nyaya Description Logics which is the most effective method to represent knowledge useful for inference and reasoning purposes. The methodology is more effective because it tackles inferences similar to the approach of human cognition. This paper also analysed the issues in existing knowledge representation formalisms. More mathematical analysis and detailed comparison of knowledge representation formalisms to promote NDL becomes our future work.

REFERENCES

Aghila G., Mahalakshmi G.S. and Geetha T.V. (2003), 'KRIL - A Knowledge representation System based on Nyaya Shastra using Extended Description Logics', VIVEK Journal, ISSN 0970-1618, Vol. 15, No. 3, pp. 3-18.

Alex B., Brachman R.J., McGuiness D.L. and Resnick L.A. (1989), 'CLASSIC: A structural data model for objects', Proceedings of 1989 ACM SIGMOD International Conference on Management of Data, pp. 59-67.

Baader F., Calvanese D., McGuinness D., Nardi D. and Schneider P.F. (2002), 'The Description Logic Handbook: Theory, Implementation and Applications', Cambridge University Press.

Brachman, R.J. (1979)On the epistemological status of semantic networks, in Findler, N., (Ed.) Associative networks: representation and use of knowledge by computers. New York: Academic Press

Brachman R.J. and Schmolze J. (1985), 'An Overview of the KL-ONE Knowledge representation System', Cognitive Science, Vol. 9, No. 2.

Bresciani E., Franconi and Tessaris S. (1995), 'Implementing and testing expressive DL, Description logics: a preliminary report', In Gerard Ellis, Robert A. Levinson, Andrew Fall and Veronica Dahl (eds.), Knowledge Retrieval, Use and Storage.

Calvanese D., Giacomo G.D. and Lenzerini M. (2002), 'Description Logics: Foundations for Class-based Knowledge representation', Roma, Italy.

Woods, W.A. (1975). What's in a link: Foundations for semantic networks. Bobrow, D.G. andCollins, A.M., Ed. Representation and Understanding: Studies in Cognitive Science. pp.35-82. New York, Academic Press.

Conceptual Graph Standard, 2002 NCITS.T2 Committee on Information Interchange and Interpretation. http://users.bestweb.net/~sowa/cg/cgstand.htm.

Concept Net (2008), http://www.conceptnet.org.

Davis, R., Shrobe, H., Szolovits, P., (1993). What is a knowledge representation. AI Magazine 14 (1), 17–33.

Davies J., Duke A. and Sure Y. (2004), 'OntoShare - An Ontology-based Knowledge Sharing System for virtual Communities of Practice', Journal of Universal Computer Science, Vol. 10, No. 3, pp. 262-283.

Fellbaum C. (1998), 'WordNet - An Electronic Lexical Database', with a preface by George Miller.

Flouris G., Plexousakis D. and Antoniou G. (2006), 'Evolving Ontology Evolution', SOFSEM 2006: Theory and Practice of Computer Science, LNCS Vol. 3831.

Lau T., Wolfman S., Domingos P. and Weld D. (2003), 'Programming by Demonstration using Version Space Algebra', Machine Learning, Vol. 53, No. 1-2, pp. 111-156.

MacGregor R. (1991), 'Inside the LOOM description classifier', SIGART Bulletin, Vol. 2, No. 3, pp. 8-92.

R. MacGregor and R. Bates (1987). The Loom Knowledge representation Language. Technical Report, USC/Information Sciences Institute.

Mahalakshmi G.S., Aghila G. and Geetha T.V. (2002), 'Multi-level Ontology representation based on Indian Logic System', 4th International Conference on South Asian Languages ICOSAL-4, India, pp. 4-15.

Mahalakshmi G.S., Anupama N., Chitra R. and Geetha T.V. (2007), Deepika – A Non-Monotonic Reasoning System Based On Indian Logic, International Conference on Soft Computing Techniques in Engineering, SOFTECH–07, Avinashilingam Univ. for Women, India, pp. 470-476.

Mahalakshmi G.S. and Geetha T.V. (2008a), 'Gurukulam-Reasoning based Learning System using Extended Description Logics', Dr. Estrella Pulido and Dr. Maria D. R-Moreno (eds.), International Journal of Computer Science and Applications (IJCSA) - Special Issue on New Trends on AI techniques for Educational Technologies, Technomathematics Research Foundation , Vol. 5, No. 2, pp. 14-32.

Mahalakshmi G.S. and Geetha T.V. (2008b), 'Reasoning and Evolution of consistent ontologies using NORM', International Journal of Artificial Intelligence : Special Issue on Theory and Applications of Soft Computing, Indian Society for Development and Environment Research, Vol. 2, No. S09, pp. 77-94.

Mahalakshmi G.S. and Geetha T.V. (2009a), 'Gautama – Ontology editor based on Nyaya Logic', Proc. of Third Indian Conference on Logic and Applications, Ramanujam R. and Sundar Sarukkai (eds.), Springer LNAI 5378, FoLLI series, pp. 234-245.

Mahalakshmi G.S. and Geetha T.V. (2009b), 'Argument Based Learning Communities', KB Systems, Spl. issue on AI in Blended Learning, Elsevier, Vol. 22, No. 4, pp. 316-323.

Matuszek C., Witbrock M., Kahlert R.C., Cabral J., Schneider D., Shah P. and Lenat D. (2005), 'Searching for Common Sense: Populating Cyc from the Web', In: Proceedings of 20th National Conference on Artificial Intelligence, Pittsburgh, Pennsylvania.

Minsky, M., (1975) A framework for representing knowledge. In: Winston, P. (Ed.), The Psychology of Computer Vision. McGraw-Hill, New York, pp. 211–277 URL: hftp://publications.ai.mit.edu/ai-publications/pdf/AIM-306.pdfi.

C. Peltason, A. Schmiedel, C. Kindermann and J. Quantz. (1989) The BACK System Revisited. Technical Report kit - report 75, Projektgruppe KIT- Fachbereich Informatik-TU Berlin.

Quantz J., Dunker G., Bergmann F. and Kener I. (1996), 'The FLEX system', Technica report, KIT-Report, Technische Universitat, Ber in, Germany,175 http://citeseer.ist.psu.edu/quantz95flex.html.

Quillian, M.R. (1968). Semantic memory. Minsky, M., Ed. Semantic Information Processing. pp.216-270. Cambridge, Massachusetts, MIT Press.

Sinha N.L. and Vidyabhusana S.C. (1930), 'The Nyāya Sutras of Gautama', Translated by S.C. Vidyabhusana, edited by Nanda Lal Sinha, Sacred Book of Hindus, Allahabad, 1930, Reprinted in 1990, Delhi: Motilal Banarsidass

Thomaz A.L., Hoffman G. and Breazeal C. (2006), 'Reinforcement Learning with Human Teachers: Understanding How People Want to Teach Robots', Proceedings of 15th IEEE International Symposium on Robot and Human Interactive Communication, pp. 352-357.

Toni C. and Daniel T. (2006), 'Learnable behavioural model for autonomous virtual agents: low-level learning', In Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems, pp. 89-96.

Vanderwende L., Kacmarcik G., Suzuki H. and Menezes A. (2005), 'MindNet: An Automatically-Created Lexical Resource', In Proceedings of HLT/EMNLP 2005 Interactive Demonstrations, Canada.

M. Vilain (1985). The restricted language architecture of a hybrid representation system. In Proceedings of IJCAI-85, Los Angeles, Ca., pp. 547-551. IJCAI.

Virupakshananda Swami (1994), 'Tarka Samgraha', Sri Ramakrishna Math, Madras.

Wada T. (1990), 'Invariable Concomitance in Navya-Nyaya', Sri Garib Dass Oriental Series No. 101, Indological and Oriental Publishers, India.

Wang Y. (2006a), 'On Concept Algebra and Knowledge representation', Proceedings of 5th IEEE International Conference on Cognitive Informatics (ICCI'06), Yao Y.Y., Shi Z.Z., Wang Y. and Kinsner W. (eds.).

Wang Y. (2006b) 'The OAR model for Knowledge representation', Proceedings of Canadian Conference on Electrical and Computer Engineering, pp. 1727-1730.

TMRF e-Book
Advanced Knowledge Based Systems:
Model, Applications & Research
(Eds. Sajja & Akerkar), Vol. 1, pp 29 – 49, 2010

Chapter 3

Knowledge Representation in Matchmaking Applications

Manish Joshi[1], Virendrakumar C. Bhavsar [2], Harold Boley[3]

Abstract The features and success of matchmaking systems largely depend on how effectively participants' product/service descriptions (profile) are modelled. We formalize the multifaceted expectations and interests of participants as 'constraints' in those profiles. We identify and define the relevant types of constraints and explicitly document challenges in matchmaking.
A Knowledge Representation Model (KRM) determines how different types of constraints are represented in any matchmaking system. We analyze a role of a KRM in a matchmaking system by reviewing seven different KRMs and listing features offered by matchmaking systems that use these KRMs. We propose a new KRM that represent various types of constraints. We describe the development of the matchmaking system that uses the proposed KRM, exemplifying its features and evaluating its performance.

1. INTRODUCTION

With the advancement of Internet technology and rapid growth in online trading, websites are becoming new virtual marketplaces. In e-marketplaces all participating sellers and buyers submit their profiles (containing descriptions of products/services offered and sought, including preferences) and wish to get a ranked list of matching profiles of other participants. Matchmaking is considered here as the process of optimally pairing up participants from two groups, typically called sellers and buyers, according to some optimality criterion (e.g. maximum similarity). Automated matchmaking is a topic of research for several years. It has extended to electronic negotiations, auctions, bartering, etc. The focus of this chapter is on automated matchmaking for e-marketplaces in a Web environment.
Constructing accurate profiles is a key task since matchmaking system's success depends, to a large extent, on the ability to represent participants' interest [16]. The word 'accurate' here refers to how effectively all expectations of participants are modelled. Hence, a knowledge representation model plays an important role in all matchmaking applications. Participant's expectations, also called

[1] Department of Computer Science, North Maharashtra University, Jalgaon, MS, India. joshmanish@gmail.com
[2] Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada. bhavsar@unb.ca
[3] National Research Council, Institute for Information Technology Fredericton, Fredericton, Canada. Harold.Boley@nrc.gc.ca

constraints, are relatively numerous and multifaceted, which make it difficult to model. In contrast to a 'process matchmaking', where expectations of a process for resources (memory/processor time) are straightforward, matchmaking in e-marketplaces is complex.

We identify and explicitly define various forms that constraints can take. We listed several different types of constraints that a matchmaking system should support. The ability of a matchmaking system to support various types of constraints determines its performance. In other words we present criteria to measure the performance of any matchmaking system. We discuss in detail complex nature of participants' profiles and expectations as the challenges in matchmaking.

A knowledge representation model is a backbone of any matchmaking system. Several KRMs have been proposed for matchmaking. Each of these KRMs has certain strengths and some limitations. We review some of the KRMs and corresponding matchmaking systems. We compare these matchmaking systems based on various aspects of matchmaking. In particular we discuss the features offered, techniques used and a matching process (algorithm) of the matchmaking systems. *Array of Features*, *Knowledge Representation Language*, *Database*, *Tree*, *Graph*, and *Hybrid* KRMs constitutes a list of KRMs that are reviewed. We provide a tabular comparative study of all features offered by matchmaking systems that are based on these KRMs.

We propose a new knowledge representation model for a Web-based matchmaking environment, which can represent all types of constraints of participants' profiles. We discuss this model in detail. We develop a matchmaking system that uses the proposed KRM and exemplify system's features and evaluate its performance. We discuss our matchmaking system in detail.

In short the objectives of the chapter are to elaborate importance and usage of a KRM in matchmaking. We also want to discuss general criteria for a matchmaking system to be a more effective system.

2. CHALLENGES IN MATCHMAKING

The complex nature of participant profiles results in some interesting and challenging aspects of matchmaking. Constraints are numerous and can be of different forms. Each constraint needs special treatment. A seller or a buyer participating in matchmaking has certain expectations regarding the results. The matchmaking systems are expected to demonstrate specific features.

2.1. Types of Constraints

A constraint is a condition on a profile facet ('feature', 'attribute'). In the literature, mostly *hard* and *soft* constraints have been defined explicitly [18, 20, 23]. We give below some of the possible variations of constraints. In subsequent sections we elaborate how our proposed model and the corresponding matchmaking system represents all these types of constraints.

a) Hard and Soft constraints

These terms reflect the relative flexibility of participants regarding the fulfilment of a constraint. In case of a soft constraint, a participant is ready to proceed with a match even if the facet value described by his/her constraint is not satisfied by the facet value of the corresponding constraint of the counterpart profile. In contrast, in case of a hard constraint a participant does not compromise with an offer/request specified in a constraint.

b) Range Value Constraints

The parties involved in matchmaking often provide a range for their offerings rather than a discrete value, e.g. 'Looking for the apartment whose rent is 500$ to 600$'. This constraint should be matched with all other counterpart constraints that offer rent in the range of 500$ to 600$.

c) Multi-Value Constraints
Participants sometimes specify multiple discrete values (disjunctive) as their choices. For example, a constraint 'I want a shared or a single apartment' should be matched with all constraints offering a shared apartment as well as with all constraints offering a single apartment.

d) Preferential Constraints
For the soft constraints of a profile, a participant may wish to indicate relative preferences among various facets. For example, consider a participant's apartment profile with rent facet preferred to facets area, type, pet-allowed. This profile can succeed in spite of low constraint satisfactions for the other facets as long as the rent constraint is highly satisfied.

e) Hidden Cost constraints
In e-business matchmaking, cost is an important facet that affects successful outcomes. Some participants (especially from the seller group) may hide facet values that could increase the cost. For example, a constraint formalizing "the rent of the apartment is $550, electricity extra", should not succeed with the constraint of a participant who seeks a rent of $550

2.2. Matchmaking Results

The process of obtaining matchmaking results and the result (intermediate as well as final) itself characterizes a few more aspects of matchmaking.

a) Compromise match effect
A concept of soft constraints leads to the notion of a compromise match. Two constraints from two profiles have a compromise match if,

i) either one or both of the constraints in a comparison are soft constraints, and
ii) the values of the facets of both the corresponding constraints do not match.

In such a case, either one or both users have to compromise with the mismatching value mentioned in the counterpart constraint. Hence we refer to it as a 'compromise match'.
As the compromise match is not an exact match, the similarity value should be reduced based on whether one or both users are compelled to compromise.
A very few matchmaking systems explicitly mention about such type of match and strategies used to resolve it.

b) Symmetric / Non-symmetric
If a matchmaking system returns identical results of matching a profile P_1 with P_2 and matching a profile P_2 with P_1, then the system is called a symmetric system, otherwise it is a non-symmetric system.
For example, let the profile P_1 has a security-deposit facet and the profile P_2 be without such a facet. A symmetric matchmaking system results in identical similarity values when P_1 is compared with P_2 and when P_2 is compared with P_1. In contrast, a non-symmetric matchmaking system results in different similarity values as a consequence of these comparisons.

c) Result Classification Categories
A participant may not be interested to have a list of all matching profiles as the result of a matchmaking process, especially when the numbers of profiles in the result are large. A participant wishes a ranked list of matching profiles preferably grouped in specific categories.

2.3. Algorithm Scalability

A matchmaking system uses a particular algorithm that complements with its KRM to produce desired results. It is essential that the algorithm should scale reasonably (in terms of computational complexity) to handle large number of participant profiles.

2.4. Domain Independence

A matchmaking system that deals with the semantics of a specific domain area should be able to adapt to other domains with minimal modifications. A matchmaking system should easily be plugged-in with other domains.

2.5. Parallelizability

With the availability of multi-core chips and high performance parallel/distributed computing, it is desirable that the algorithms used for matchmaking can be ported to suit to the distributed environment.

Let's analyze various KRMs and their corresponding matchmaking applications in the next section by discussing features offered by these systems.

3. ANALYSIS OF VARIOUS KRMs

Many matchmaking systems are available. A matchmaking system uses some KRM to represent participant profiles. We discuss various KRMs and the matchmaking systems developed using these KRMs in the following subsections.

3.1. Array (Vector) of Features

This is a basic knowledge representation model used in early matchmaking systems. User profiles are stored either in the form of document or in a database or in a file using XML. The keywords extracted from the documents are used for matchmaking among the documents. A typical Information Retrieval (IR) methodology is used as the basis of matchmaking. Let's briefly describe structure of two matchmaking systems.

a) COINS system
Kuokka and Harada [12] presented one of the early matchmaking systems, COINS (COmmon INterest Seeker). It uses a distance measure of information retrieval technique to carry out matchmaking on free text. The COINS system converts the free text document into a document vector, which is used later for processing. It uses SMART [19] information retrieval system to process and match free text and document vectors.

b) GRAPPA system
Veit et al. [23] have developed the Generic Request Architecture for Passive Provider Agent (GRAPPA) matchmaking framework and library system in 2001. The matchmaking engine accepts a set of offers and requests as an input. A distance function recursively computes the distance values (between 0 and 1) for different profile subtypes. Based on these distance values, the system returns a ranked list of the best 'k' candidate profiles matching to a given profile. The system used a typical vector space model to identify distances among the keywords occurring in a document. The system maintains a document vector and calculates a term-frequency-inverse-document-frequency factor

(tf-idf factor), which is used in further processing to determine the similarity between two documents. The GRAPPA system uses XML to describe user profiles.

3.2. Database

A matchmaking system that uses a database to represent knowledge is generally developed for a specific domain. The domain specific information is organised appropriately in the database. Matchmaking in this case basically relies on IR based techniques. The ontological part is embodied with the system to obtain semantically acceptable results of matchmaking.
Liesbeth et al. [14] developed a matchmaking system to respond to a learner's request by matching profiles of other learners who wish to share knowledge, by determining their content competence, sharing competence, eligibility and availability.
The database system is used to store, learning contents that are organized in courses and user (learner) profiles. A user profile consists of completed courses, current courses, activities, calendar and other information.
A leaner input a query to the system using the request module interface and the query data is stored in the database. A Latent Semantic Analyser (LSA) maps the content question on the available documents in the database to generate a list of all suitable matching resources, i.e. other learners who are content competent, sharing competent, eligible, available, etc.

3.3. Tree

Some researchers have proposed the use of a tree structure to represent the knowledge. A basic tree structure is used by Islam et al. [10]. They proposed a matchmaking framework to identify the set of matching resources for a job, from a large collection of resources in a distributed environment.
Bhavsar et al. [3] developed a matchmaking system that uses node labelled, arc labelled, arc weighted trees to represent knowledge. Nodes of a tree are used to represent the concept and branches represent the relationship. A 'weight' is assigned to an arc to signify the importance of corresponding relationship in matchmaking. A recursive bottom up approach based customized algorithm is designed to evaluate similarity among such trees and list of matching profiles is generated.

3.4. Graph

Like in a tree structure, the nodes of a graph are used to represent concepts and edges of a graph represents relationship among these concepts. Mohaghegh et al. [15] proposed a matchmaking system in the domain of online recruitment. It compares available resumes with a special kind of skills ontology in different skills and relationship among skills are represented as nodes and edges of a graph. Similarities among skills are described with the help of a graph structure. Advertisements and resumes are attached to appropriate nodes of the graph based on the contents of documents. A path-set is used to evaluate score between a given advertisement and a resume. A specific function calculates score values between an advertisement and all resumes. Ranking of resumes is provided as a result of the matchmaking.
In the IMPACT system [21], Yellow Pages Servers play the role of matchmaking agents. Offers and requests are described in a simple data structure that represents a service by a verb and one or two nouns. The matchmaking process finds k-nearest service names, and agents who provide these services. All these service names are within a specified distance d. The weights on edge of the graph reflect distances between a set of verbs and nouns, which are represented as nodes of a graph.
Bellur and Kulkarni [2] used a variation of graph, *Bipartite graph*, for matchmaking of web services.

3.5. Knowledge Representation Languages

Knowledge Representation (KR) languages are used to represent the concept definitions of an application domain in a structured and formally well-understood way [1]. Matchmaking systems based on KR languages emphasize the semantics, in contrast to earlier matchmaking systems that focus on the frequency of keywords. Several matchmaking systems are developed that use description logic to model the domain knowledge. A semantic reasoner is used for matchmaking in some of the systems. In other systems customised algorithms have been developed for matchmaking.
Some of the matchmaking systems that use knowledge representation languages are described below.

a) LARKS based system
Sycara et al. [22] have proposed an agent capability description language called LARKS (Language for Advertisement and Request for Knowledge Sharing), that is used in a Retsina multi-agent infrastructure framework. A matchmaking process is carried out at five possible levels, namely, context matching, profile comparison, similarity matching, signature matching and semantic matching.
A standard technique of the Information Retrieval is used for the syntactic matching that includes profile comparison, similarity matching, and signature matching. Whereas, the semantic matchmaking is achieved using a local ontology, written in a specific concept language ITL.
The system identifies three types of matches among profiles. An exact match is the most accurate type of match, plug-in match is a less accurate, and relaxed match is the least accurate type of match [22].

b) Description Logic based NeoClassic Reasoner
Di Noia et al. [5] developed a system that facilitates semantics-based matchmaking. It uses a description logic (DL) based framework to represent knowledge. They have proposed a three-way classification of matches as - exact, potential and partial. The system provides a numeric score that indicates how far the demand is with reference to the supplies. These numeric scores are used to rank the results. An apartment-rental ontology based case study is discussed in the paper.

c) Semantic Web language based System
Li and Horrocks [13] developed a service matchmaking system that uses DAML-S (OWL-S) service description language and ontology. OWL-S uses a set of mark-up language constructs to describe the properties and capabilities to represents web services in unambiguous, computer-interpretable form. A prototype matchmaking that uses a DL reasoner (RACER) [8] is developed. The prototype system matches service advertisements and requests using ontology based service descriptions semantics. The system proposes the concept of *degree of match*, which is used to classify the match results in five different classes. The query is divided into volatile query and persistent query based on the duration for which it remains active.
One of the earliest matchmaking systems proposed by Finin et al. [6] was based on the KQML and used a Rule Based Approach for matchmaking.
Hoffner et al. [9] proposes matchmaking as a starting point of negotiations between a demand and a supply in a peer-to-peer way. The matchmaking engine (MME) handles supplies/demands as properties and rules. The properties are name-value pairs constructed using an extension of the Corba Trading service language. Rules are constructed using a generic script language. The matching is accomplished by comparing properties and verifying rules.
Based on Semantic Web concepts and ontologies developed, Gupta et al. [7] claim to improve the performance of web service query matchmaking.
All these matchmaking systems use knowledge representation languages to represent user's profile.

3.6. Hybrid

A combination of different techniques is used to represent user information. Ragone et al. [18] propose a semantic matchmaking approach that mixes various knowledge representation technologies. It uses a combination of DLR-Lite, fuzzy rules, and utility theory to represent users profiles. In particular fuzzy rules are used to represent the concept of hard *constraints* and *soft* constraints. Sellers and buyers can assign utility values to indicate preferences among the constraints. The system uses vague Datalog rules that can assign appropriate scores depending upon the values in the profiles so that fuzzy descriptions like 'cheap cars' can be incorporated. The system returns top *k* matches and ranks them based on the score that is computed using the datalog rules, utility values assigned to the constraints, etc.

3.7. Other

A 'one-input transition system' based model, slightly similar to Deterministic Finite Automaton (DFA), proposed by Çelebi et al. [4] and a Neural Network [24] are also used as KR models for 'process matchmaking'. As mentioned earlier, we do not explore these KRM based systems as we wish to compare matchmaking systems used in e-marketplaces.
A fuzzy linguistic approach is used to model and match buyer's preference with products [17]. But for the process of matchmaking the system considers only two features of the products, which make the profile far simple.
All matchmaking systems based on the principles of different KRMs, are compared using some of the characteristics listed in earlier section. Table 2 at the end of chapter shows the cross-dimensional analysis of such systems.

4. A PROPOSED KRM

We propose to represent a participant profile as a set of constraints, such that $P = \{C_1, C_2, C_3, ...C_m\}$. Each constraint is a quadruple $C_i =< a, d, f, p >$, where a is an attribute, d is a set of values used to describe an attribute, flexibility that determines whether a constraint is a soft or a hard constraint is indicated by f and p is the priority of a constraint. All elements of a constraint are described below.

Attribute (a)
An *attribute* represents the facet. For example, if a participant has a constraint 'need 4 bedrooms', then the attribute of this constraint is 'bedrooms'. This field always has an alphabetical value. Let A be the domain of a, such that $a \in A$. An illustrative list of a set A members is shown in Table 1.

Description (d)
Description represents a set of values that can be assigned to an attribute of a constraint. In the example of 'need 4 bedrooms', the attribute 'bedrooms' of the constraint has the description value '4'. Let D be the domain of d. $d \subset D$. D contains all possible member values that a description set can have. D contains alphabetical strings that describe an attribute, or numerical values that can be assigned to an attribute, or a combination of both, or a range value having a format like $num1 \cdots num2$ such that $num1, num2 \in R$. A sample of a set D is also shown in Table 1.
Sometimes a party assigns more than one value to describe an attribute of a constraint, for example, 'looking for a shared apartment or bachelor apartment'. As the description (d) is a set of values, it can represent *multi-value constraints*. Hence for the above example, the constraint is represented as

<type, {sharedApartment, BachelorApartment}, f, p>. The set of constraints 'rent is 1500', 'available from September-1', and 'pets should be allowed' can be represented as <rent, {1500}, f, p)>, <availableDate, {Sept-01}, f, p)> and <pets, {allowed}, f, p> respectively. In these examples, we have not specified any values of f and p for the constraints.

Consider a user who asks for a '2 or 3 bedroom apartment'. In this case, the attribute 'bedrooms' have a description value that can be represented as a set of 'multiple values' or a 'range'. Hence <bedrooms, {2, 3}, f, p> and <bedrooms, {2⋯3}, f, p> are both valid representations and have identical meanings. Figure 1 shows a rent constraint that has a range description.

Table 1: An example Attribute (A) set and Description (D) set.

Attribute Set A	Description Set D
Area	downtown, riverside, north
Available	September-01, Fall, Summer
Bedrooms	3, 1⋯3
Cats	Allowed, No
Dogs	Not-allowed
Kids	2, No-kids
Laundry	Coin-operated, Yes
Pets	Yes, No
Rent	500, 350, 1000⋯1200
Smoking	Not-Permitted, Allowed
Type	Apartment, SharedHouse

Flexibility (f)

Flexibility indicates whether the constraint is a hard or a soft constraint. $f \in F$, where $F = \{No, Yes\}$. A 'No' value of f (i.e. no flexibility) indicates a rigidness of the constraint, whereas a value 'Yes' represents a soft constraint. A soft constraint is matched with any value of the counterpart as a *compromise match*. A constraint specification provided by a buyer as 'house rent must be 500' indicates a hard constraint and is represented as <rent, {500}, No, p>. A constraint description 'Smoking is not allowed, but can smoke in balcony', represents a soft constraint. It can be represented as <Smoking, {Not allowed}, Yes, p>.

Priority (p)

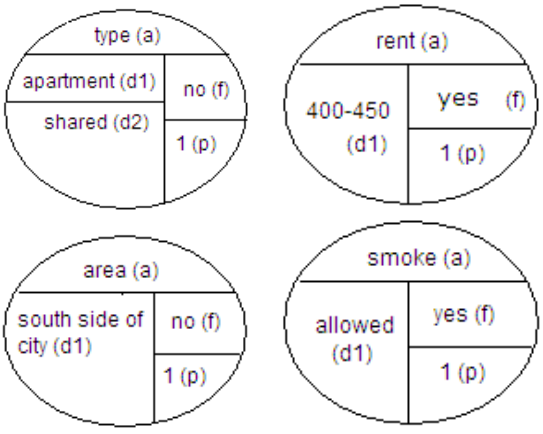
The *priority* describes the relative priority of soft constraints among other soft constraints, in a profile. The value of p can be any real value grater than 0. $p \in R$. All soft constraints are initialized with the priority values of 1. The priority values for all soft constraints are set automatically to match the preferences indicated by participants.

For example, if a buyer specifies that pets allowed facet is more important to him than all remaining facets, then priority value for this constraint is set to a value grater than 1. The constraint is represented as <pets, {allowed}, No, 1.1>, and all remaining constraints will have p values as 1. Note that, the value of flexibility in this example, is 'No', indicating a hard constraint. These priority values ultimately used to rank the service represented by the facet.

The Figures 1 and 2 illustrate how a buyer's (a tenant's) profile and a seller's (a computer owner's) profile can be represented in our model. The description of the participants profiles is followed by a node representation (Figures 1(a), 2(a)) and a quadruple representation (Figures 1(b), 2(b)).

Profile-1 – Tenant (Buyer)

I am a mature student looking for an affordable shared or single apartment on the south side of Fredericton for September. Finishing up my last year at UNB, I smoke but can adjust with non-smoking apartment. rent – 400 to 450. Please contact if anything is available, thanks!



(a) Constraints as nodes

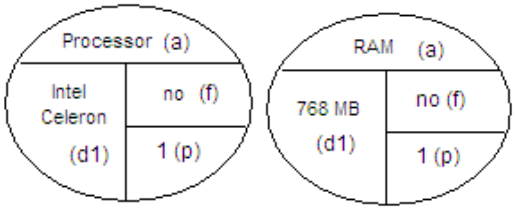
```
<type, {apartment, shared}, No, 1>
<rent, {400···450}, Yes, 1>
<area, {South side}, No, 1>
<smoke, {allowed}, Yes, 1>
<available, {Sept-01}, No, 1>
```

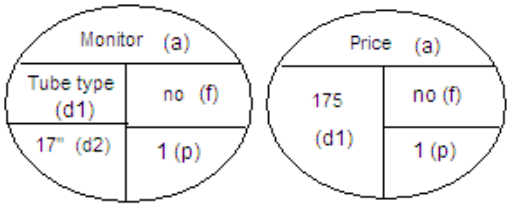
(b) Constraints as quadruples

Figure 1. Representation of the constraints of a Buyer

Profile-2 – Computer Owner (Seller)

Intel Celeron processor at 1.7 Ghz, 768 MB RAM, 40 GB hard drive, Nvidia GeForce FX5700LE 256 mb video card, floppy, CD burner, 17" tube type monitor, Windows XP Pro installed (no disk). Completely configured and ready for high speed internet connection, includes AVG anti-virus. Works great!





(a) Constraints as nodes

```
<Processor, {Intel Celeron}, No, 1>
<Speed, {1.7 GHz}, No, 1>
<RAM, {768 MB}, No, 1>
<HDD, {40 GB}, No), 1>
<VideoCard, {256 MB}, No, 1>
<OpticalDevice, {CD Writer}, No, 1>
<Monitor, {17 inch, tube type}, No, 1>
<OS, {Windows XP}, No, 1>
<Internet, {high speed}, No, 1>
<Softwares, {AVG antivirus}, No, 1>
<Price, {175}, No, 1>
<Type, {personal computer}, No, 1>
```

(b) Constraints as quadruples

Figure 2. Representation of the constraints of a Seller

Appendix-1 shows sample Seller's, Buyer's profiles and matchmaking results obtained for these sample profiles.

The next section elaborates algorithmic steps to compute similarity value between any two profiles.

5. A MATCHMAKING ALGORITHM

The similarity value between any two profiles is defined as a function of attribute, description, flexibility and priority values of all constraints from both profiles. For any two profiles P_x and P_y, where P_x has m constraints and P_y has n constraints, a similarity value is given by,

$$Sim(P_x, P_y) = \prod_{i=1 \text{ to } m,\ j=1 \text{ to } n} S(C_i, C_j) \quad (1)$$

where the function $S(C_i, C_j)$ calculates an intermediate similarity value using steps given in the algorithm below.

An attribute, a description, a flexibility and a priority value of a constraint, are represented using notations $C_i.a$, $C_i.d$, $C_i.f$, and $C_i.p$ respectively for a constraint C_i.

```
1:  if (Ci.a=Cj.a) then
```

```
2:       if (Ci.d = Cj.d)then
3:           return S(Ci , Cj) =  Ci .p  ×  Cj.p
4:       else
5:           if (Ci.f = No) AND (Cj.f = No) then
6:              return S(Ci,Cj)= Ci.p × Cj.p ×
                relativeDifference(Ci.d ,Cj.d)
7:           elseif (Ci.f = Yes) AND (Cj.f = Yes)
8:              return S(Ci,Cj)= Ci.p × Cj.p ×β
9:           else
10:             return S(Ci,Cj)= Ci.p × Cj.p ×α
11:      move on to next Ci and Cj
12:  if (Ci.a < Cj.a) then
13:          return S(Ci, Cj)= Omission Penalty
14:          move on to next Ci
15:  if (Ci.a > Cj.a) then
16:          return S(Ci, Cj)= Omission Penalty
17:          move on to next Cj
```

The algorithm compares two constraints of two profiles. If the attributes of both the constraints are same then an intermediate similarity value is calculated by checking the description values. If the description values are not same then an intermediate similarity value is calculated by considering the flexibility of the constraints. When hard constraints in two profiles do not match, instead of reducing a similarity value immediately to zero, we compute relative difference between the two corresponding description values of these attributes. A routine *relativeDifference* computes relative difference, which is later used to calculate a similarity value. Note that for numeric and alphabetical values of d, separate routines are required to obtain relative differences. We make sure that an intermediate similarity value for such constraints is reduced substantially.

Numeric relative difference between profiles having rent as 500 and 700 (where numeric difference is 200) is not the same as profiles having rent as 1500 and 1700. Rather the first difference (i.e. between 500 and 700) is relatively greater than the second.

The parameters α and β are *compromise count factors* used in case of compromise match and its usage is elaborated in next section.

Appendix-2 shows an example of how matchmaking algorithms results appropriate similarity values when profiles are matched with each other.

6. 'HUNT FOR TUNE' FEATURES

In the previous section, it is shown how the proposed model represents multifaceted constraints. In this section, we describe additional features supported by the 'Hunt For Tune' matchmaking system that is based on the proposed KRM.

6.1. Preferential Constraints

Our model facilitates participants to indicate the relative importance among soft constraints, if any. For example, a participant can indicate $facet_1 > facet_5 > facet_3$ using an interface and appropriate priority values are assigned to the corresponding constraints. Figure 3 shows screenshot of the GUI of the 'Hunt For Tune' matchmaking system.

Each constraint is initialized with a priority value 1 and it is gradually incremented after user clicks on '+' button placed beside the priority value of a facet (see Figure 3). This interface allows participant to input his/her constraints. Using this interface the participant can indicate preferences among soft facets easily. In Figure 3, the preference of an 'availableDate' facet is set to 1.1, while for all other soft constraints the priority is set to 1.

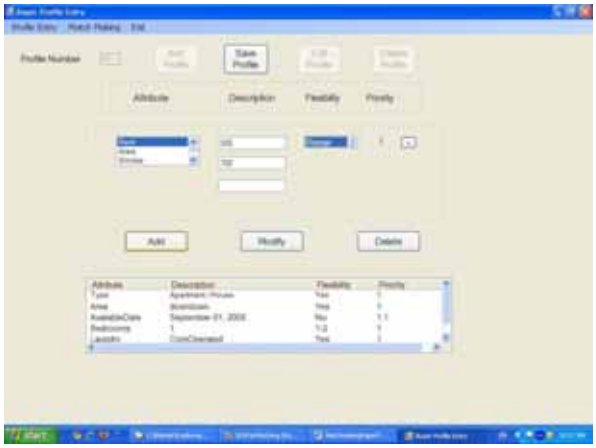


Figure 3. Screenshot of the GUI for profile entry

6.2 Hidden Cost Constraints

We propose that a profile with a hidden cost constraint should be penalized in the process of matchmaking. Hence a constraint, which carries hidden cost, has to bear the hidden cost penalty.

In our matchmaking system, we reduce the priority value of the hidden cost constraint to 0.9. This value is less than the priority values of all remaining constraints (all other constraints have priority values of at least 1).

Due to the penalty, in term of reduction in priority, similarity value of a profile that contains hidden cost constraint, will be less than a profile that do not have hidden cost constraint.

6.3 Symmetry/Non-symmetry

We introduce a parameter *omission penalty*, and its value can be set by a user. This parameter value is used to reduce the resulting similarity value while matchmaking, for each constraint that is present in the seller's profile but missing from the buyer's profile; or vice versa.

If the value of an omission penalty is set to 0, the system shows characteristics of a symmetric matchmaking system, i.e. $Sim(P_x, P_y) = Sim(P_y, P_x)$. For any other value of omission penalty such that $0 < \text{omission penalty} \leq 1$, the matchmaking system exhibits non-symmetric characteristics from buyers or sellers viewpoint.

6.4 Compromise Match Effect

A compromise match is not an exact match hence a similarity value between corresponding profiles should be reduced. In our matchmaking system, when there is a compromise match between two constraints, an intermediate similarity value (given by the function *S* in equation 1) is reduced by a certain factor. Consider an example of a soft constraint by a seller, "Prefer a non-smoker but ready to deal with a smoker" and a buyer's soft constraint as "I am looking an apartment where smoking is

allowed but ready to rent a non-smoking apartment too". These two constraints have a compromise match. As both of the participants are ready to compromise with their preferred choices, it is likely that these two participants can reach an agreement. Hence a similarity value in case of a compromise match is influenced by the count (*compromise count*) of participants (one or both) willing to compromise.

We propose two *compromise count factors*, α and β to reduce a similarity value, in case of a compromise match. The values of α and β are set to less than 1. An intermediate similarity value is multiplied by these factors to obtain an expected reduction in a similarity value.

If a compromise count is one, then there are relatively fewer chances of an agreement as only one participant is ready to compromise. The factor α represents this case, while the factor β is used when compromise count is two.

We set the values of α and β such that a higher similarity value shall be resulted for a compromise match where both participants are ready to compromise and a lower similarity value shall be resulted if only one participant is ready to compromise.

6.5 Result Classification Categories

A user desires to obtain a list of matching profiles classified among categories and ranked within the categories.

We propose following six categories for matchmaking results.

1. Matching all hard constraints and matching all soft constraints.
2. Matching all hard constraints and matching some soft constraints and absence of remaining soft constraints in counterpart profile (leading to further action like – inquiring).
3. Matching all hard constraints and absence of all soft constraints in counterpart profile.
4. Matching all hard constraints, some compromise match, and some missing constraints.
   A. Compromise match constraints where both parties willing to compromise.
   B. Compromise match constraints where only one party is willing to compromise.
5. Not matching hard constraints and the margin of difference in description values is less.
6. Not matching hard constraints and the margin of difference in description values is high.

6.6 Scalability

The KRM uses simple set of nodes to capture key information associated with the participant profiles. It avoids overhead of building complex data structures like graph and tree. The algorithm compares two profiles and generates the similarity value in linear time. Hence we could expect this approach to generate results in satisfactory amount of time even for large number of profiles. The algorithm for matchmaking can easily be converted to suit for a distributed/parallel computing.

6.7 Domain Independence

The KRM is totally independent of domain and can be applicable to many domains. This KRM describes a general technique to capture essence of any type of constraint. It has the provision to capture various options offered/demanded by participant in any constraint.

In order to be useful in any domain a specific ontology for that domain shall be required. The semantic relative difference routine used in the algorithm and other features largely depends upon domain knowledge.

6.8 Automatic categorization

As the nodes are created by considering attribute values and description values of constraints among profiles, the KRM can be programmed to count and categorize profiles based on these values. A more descriptive categorization shall be available after processing of all profiles.

7. EVALUATION

We have obtained results of the matchmaking system developed using our KRM for a house rental domain. Our system supports all the types of constraints discussed in 'Challenges in Matchmaking' section. The system generates an appropriate list of similarities among profiles. The system facilitates users to determine the ranking of matching profiles by tuning the values of parameters like the omission penalty and the compromise count factors. It would be interesting to study the effect of change in parameter values on matchmaking result classification.

8. CONCLUSION

We discuss a role of a KRM in automated matchmaking. We enlist several features that matchmaking systems should exhibit. We used these features to review several KRMs and their corresponding matchmaking systems.
We have proposed a new model for knowledge representation that represents complex constraints of users participating in automated matchmaking. We discuss how our system offers many additional features as compared to other matchmaking systems.

ACKNOWLEDGEMENTS

This research work was partially supported by a post-doctoral fellowship from ACEnet, Canada (www.ace-net.ca) and a research grant of Dr. Bhavsar from the Natural Sciences and Engineering Research Council of Canada (NSERC). The research work was partly carried out when Dr. Bhavsar was visiting the International Institute of Information Technology, Pune, India.

REFERENCES

[1] F. Baader, D. Calvanese, D. Mcguinness et al., " The Description Logic Handbook: Theory, Implementation and Applications," Cambridge University Press, Cambridge, MA, 2003.

[2] U. Bellur and R. Kulkarni, "Improved matchmaking algorithm for semantic web services based on bipartite graph matching," in *IEEE International Conference on Web Services,* 2007,

[3] V. C. Bhavsar, H. Boley and Y. Lu, "A Weighted-Tree Similarity Algorithm for Multi-Agent Systems in e-Business Environments," *Computational Intelligence,* vol. 20, pp. 584-602, 2004.

[4] R. Çelebi, H. Ellezer, C. Baylam, I. Cereci and H. Kılıç, "Process Matchmaking on a P2P Environment," *Procceding of International Conference on Web Intelligence and Intelligent Agent Technology,* pp. 463-466, 2006.

[5] T. Di Noia, E. Di Sciascio, F. M. Donini and M. Mongiello, "A System for Principled Matchmaking in an Electronic Marketplace," *International Journal of Electronic Commerce,* vol. 8, pp. 9-37, 2004.

[6] T. Finin, R. Fritzson, D. Mckay and R. McEntire, "KQML as an agent communication language." in *Third International Conference on Information and Knowledge Management,* 1994, pp. 456-463.

[7] C. Gupta, R. Bhowmik, R. H. Michael, M. Govindaraju and W. Meng, "Improving performance of web services query matchmaking with automated knowledge acquisition," in *International Conference on Web Intelligence,* 2007, pp. 559-563.

[8] V. Haarslev and R. Moller, "RACER System Description," *Proceeding of the International Joint Conference on Automated Reasoning (IJCAR 2001) Lecture Notes in Artificial Intelligence,* vol. 2083, pp. 701-705, 2001.

[9] Y. Hoffner, A. Schade, C. Facciorusso and S. Field, "Negotiation Protocol Characterisation and Mechanism for Virtual Markets and Enterprises" *Collaborative Business Ecosystems and Virtual Enterprises,* 2002.

[10] M. R. Islam, M. Z. Islam and L. Nazia, "A Tree-based Approach to Matchmaking Algorithms for Resource Discovery," *International Journal of Network Management,* 2008.

[11] M. Joshi, V. Bhavsar and H. Boley, "A Knowledge Representation Model for Match-Making Systems in e-Marketplaces," Proc. of the 11th International Conference on e-Commerce (ICEC 2009), August 12-15, 2009, Taipei, Taiwan, pp. 362-365, 2009.

[12] D. Kuokka and L. Harada, "Integrating Information via Matchmaking," *Journal of Intelligent Information Systems,* vol. 6, pp. 261-279, 1996.

[13] L. Li and I. Horrocks, "A Software Framework for Matchmaking Based on Semantic Web Technology," *International Journal of Electronic Commerce,* vol. 8, pp. 39-60, 2004.

[14] K. Liesbeth, P. Rosmalen, P. Sloep, F. Brouns, M. Koné and R. Koper, "Matchmaking in Learning Networks: Bringing Learners Together for Knowledge Sharing," *The Netherlands Interactive Learning Environments,* vol. 15, pp. 117-126, 2007.

[15] S. Mohaghegh and M. R. Razzazi, "An Ontology Driven Matchmaking Process," *World Automation Congress,* vol. 16, pp. 248-253, 28 June - 1 July 2004. 2004.

[16] M. Montaner, B. Lopez and Josep LLuis, De La Rosa, "A Taxonomy of Recommender Agents on the Internet," *Artificial Intelligence Review,* vol. 19, pp. 285-330, 2003.

[17] A. C. Ojha and S. K. Pradhan, "Fuzzy Linguistic Approach to Matchmaking in E-Commerce," *International Conference on Information Technology Proceedings,* 2005.

[18] A. Ragone, U. Straccia, T. Di Noia, E. Di Sciascio and F. M. Donini, "Vague Knowledge Bases for Matchmaking in P2P E-Marketplaces," *ESWC,* 2007.

[19] G. Salton, *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer.* Addison-Wesley, 1989,

[20] M. Ströbel and M. Stolze, "A Matchmaking Component for the Discovery of Agreement and Negotiation Spaces in Electronic Markets," *Group Decision and Negotiation,* vol. 11, pp. 165-181, 2002.

[21] V. S. Subrahmanian, P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Ozcan and R. Ross, "Heterogenous Agent Systems," *MIT Press,* 2000.

[22] K. Sycara, S. Widoff , M. Klusch and J. Lu, "Larks: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace," *Autonomous Agents and Multi-Agent Systems,* vol. 5, pp. 173-203, 2002.

[23] D. Veit, J. P. Müller and C. Weinhardt, "Multidimensional Matchmaking for Electronic Markets," *International Journal of Applied Artificial Intelligence,* vol. 16, pp. 853-869, 2002.

[24] Z. Zhang and C. Zhang, "An Improvement to Matchmaking Algorithms for Middle Agents," *Procceding of International Conference on Autonomous Agents*, pp. 1340-1347, 2002.

APPENDIX-1

Each of the profiles P-1 to P-6 is matched with profiles P-8 to P-14 to obtain similarity values. All these profiles are obtained from an online free local classifieds service available at 'http://fredericton.kijiji.ca'.
Only those matching profiles are displayed in the result where similarity value is non-zero.

House Owner's Profile		
P-1 <bedrooms,{4}, No, 1> <laundry,{yes}, No, 1> <lease,{1-year}, No, 1> <rent, {1700}, No, 1> <type,{apartment},No,1>	**P-2** <available,{Sept-1},No,1 <pets, {no}, No, 1> <rent, {395}, No,1> <smoke,{no}, No,1> <type,{bachelor},No, 1>	**P-3** <available,{Sept-1},No,1 <bedrooms,{3},No, 1> <rent, {600-900}, No,1> <security,{700},No, 1> <type,{apartment},No,1>
P-4 <bedrooms,{1},No,1> <parking,{1}, No,1> <rent, {625}, No, 1> <type,{apartment},No,1>	**P-5** <rent, {300},No, 1> <type, {room}, No, 0.99>	**P-6** <available,{Aug-1},No,1 <bedrooms,{2}, No, 1> <laundry,{yes}, No, 1> <parking,{2}, No, 1> <rent, {900}, No, 1> <type,{apartment},No,1>

Buyer – Tenant's Profile		
P-8 <available,{Sept-1}, No, 1> <bedrooms,{1}, No, 1> <rent, {100-400}, No, 1> <type,{bachelor, room}, No,1>	**P-9** <available, {Sept-1}, No, 1> <rent, {375}, No, 1> <type,{room},No, 1>	**P-11** <bedrooms,{ 2}, No, 1> <kids,{yes}, No, 1> <pets, {yes}, No, 1> <rent, {500}, No, 1> <type,{apartment},Yes,1>
P-12 <available,{Sept-1},No, 1> <parking,{1}, Yes, 1> <rent,{500}, No, 1> <type,{bachelor}, No, 1>	**P-13** <pets, {yes}, No, 1> <rent, {0},Yes, 1> <type,{room}, Yes, 1>	**P-14** <area,{downtown},No, 1> <available, {Sept-1}, No,1> <bedrooms,{2}, No, 1> <kids, {no}, No, 1> <laundry,{yes},No, 1> <pets,{yes}, No, 1> <rent, {800}, Yes, 1> <type,{apartment},No,1>

Matchmaking Results –

Similarity value - profile 1 Vs. profile 13 is -->0.9412
Similarity value - profile 1 Vs. profile 14 is -->0.397635
Similarity value - profile 1 Vs. profile 11 is -->0.1396
Similarity value - profile 2 Vs. profile 12 is -->0.985
Similarity value - profile 2 Vs. profile 8 is -->0.9652
Similarity value - profile 3 Vs. profile 14 is -->0.4315

Similarity value - profile 4 Vs. profile 14 is -->0.9506
Similarity value - profile 4 Vs. profile 13 is -->0.946
Similarity value - profile 4 Vs. profile 11 is -->0.4703
Similarity value - profile 5 Vs. profile 9 is -->0.995
Similarity value - profile 5 Vs. profile 13 is -->0.9751
Similarity value - profile 5 Vs. profile 8 is -->0.9702
Similarity value - profile 5 Vs. profile 11 is -->0.9653
Similarity value - profile 6 Vs. profile 13 is -->0.93639
Similarity value - profile 6 Vs. profile 11 is -->0.4268

APPENDIX-2

Following cases elaborate how the matchmaking algorithm calculates similarity values. Profile P1 is matched with P8, P13, P14 and P11 respectively.

P-1: <bedrooms,{4}, No, 1> <laundry,{yes}, No, 1> <lease,{1-year}, No, 1> <rent, {1700}, No, 1> <type,{apartment},No,1>

Case 1: P-1 Vs P8

P-8 : <available,{Sept-1}, No, 1> <bedrooms,{1}, No, 1> <rent, {100-400}, No, 1> <type,{bachelor, room}, No,1>

Here we have a mismatch of 3 Hard constraints. The attributes bedrooms, rent and type are hard constraints and description values of these two profiles mismatch. Hence the **Similarity Value: 0.0**

Case 2: P-1 Vs P13

P-13 : <pets, {yes}, No, 1> <rent, {0},Yes, 1> <type,{room}, Yes, 1>

Some attributes like bedrooms, laundry, lease from P-1 are not present in P-13 and attribute pets from P-13 is missing in P-1. But rent and type are soft constraint in one profile (P-13). Hence we have two Compromised matches of soft constraints. Hence the **Similarity Value: 0.9412**

Case 3: P-1 Vs P14

P-14 : <area,{downtown},No, 1> <available, {Sept-1}, No,1> <bedrooms,{2}, No, 1> <kids, {no}, No, 1> <laundry,{yes},No, 1> <pets,{yes}, No, 1> <rent, {800}, Yes, 1> <type,{apartment},No,1>

The attributes bedrooms and type are hard constraints and description values of these two profiles mismatch. It is a mismatch of 2 Hard constraints and there are two compromised matches (laundry and rent). **Similarity Value: 0.397635**

Case 4: P-1 Vs P11

P-11: <bedrooms,{ 2}, No, 1> <kids,{yes}, No, 1> <pets, {yes}, No, 1> <rent, {500}, No, 1> <type,{apartment},Yes,1>

The attributes bedrooms and rent are hard constraints and description values of these two profiles mismatch. So in total there is a mismatch of 2 Hard constraints but only one compromised match (type). Hence similarity value of this match is less than that of case 3. **Similarity Value:0.1396**

Table-2: Cross dimensional analysis of various KRMs used for matchmaking systems

A KRM	Matchmaking System	Types of Constraints supported	Result Classification Categories	Algorithm Details Matching Process
Array	COIN (1996) [12]	None	• Uses a distance measure of IR. • Returns matching document vector, including the name of the document.	IR based process – No Ontologies used The SMART [19] information retrieval system is used, to process and match free text and document vectors. A local concept corpus is maintained but it does not implement notion of semantic matchmaking,
	GRAPPA (2001) [23]	Hard / Soft Alternate Value	• Identifies k nearest requests based on distance function	Parallel amenable, IR based matching Profiles represented in XML Do not support N:M matching
Knowledge Representation Languages	LARKS based RETSINA System (2002) [22]	Range Value Alternate Value	• Exact, plug-in, relaxed. • Generate ranked list based on similarity value.	Performs syntactic and semantic matching. Uses IR technique to compare profiles. Concepts are stored using Information Terminological Language (ITL). Different matching modes apply different combination of filters for matchmaking
	Description Logic based NeoClassic (2004) [5]	Range Value Preferential (Weights are used to increase relevance of concept) Alternate Value	• Exact, Potential, Partial • Ranking of matching profiles	Rankpartial and rankpotential algorithm produce a distance based rank when a demand is matched with several supplies. Ontologies are used
	Web Service Technology OWL-S (2004) [13]	Range Value Alternate Value	• Exact, plug-in, subsume, intersection, disjoint.	A Description Logic reasoner, RACER, is used to compute semantic matches. OWL-S service ontology is used for service description.
Database	Match making in Learning Networks	None	• List of all resources matching to the requested service are populated. • Matching criteria are- sharing and content competence, availability,	Especially used for matching resources in Learning Network. Database stores details of learning contents, learner information and available resources.

Table-2: Cross dimensional analysis of various KRMs used for matchmaking systems (Continue…)

A KRM	Matchmaking System	Types of Constraints supported	Result Classification Categories	Algorithm Details Matching Process
Tree	A Weighted-Tree (2004) [3]	Range Value Preferential Alternate Value	• A list of matching profiles is displayed.	Information of demand and supply is stored in node-labeled, arc-labeled, arc-weighted tree. A tree matching algorithm computes similarity.
Graph	An Ontology Driven Matchmaking Process (2004) [15]	Range Value - Not Clear Preferential - Not clear Alternate Value - Not clear	• A path set is determined after matching resumes with advertisement. • Weights assigned to relationships and nodes, are used to calculate scores. • System returns k best resumes.	Skill ontology is maintained as a graph. Nodes represent skills (hard and soft) and edge represents inheritance relationship between nodes.
Hybrid Combination of Description Languages, Fuzzy Rules and Utility theory	Vague Knowledge Bases for Matchmaking in P2P E-Marketplaces (2007) [18]	Hard / Soft Range Value (Discrete values) Preferential (Use of utility theory) Alternate Value	• Returns top k matching profiles based on the score.	Use of Fuzzy predicates supports vague rules Semantic matchmaking
Proposed (set of Nodes)	Hunt ForTune matchmaking system (2009) [11]	Hard / Soft Range Value Preferential Alternate Value Hidden Cost	• Similarity value • Six categories	Scalable algorithm Amenable for Parallelization Parameter supported

TMRF e-Book
Advanced Knowledge Based Systems:
Model, Applications & Research
(Eds. Sajja & Akerkar), Vol. 1, pp 50 – 73, 2010

Chapter 4

Diagnostic Expert Systems: From Expert's Knowledge to Real-Time Systems

C. Angeli [1]

Abstract

This Chapter presents the evolution of the expert systems paradigm for fault diagnosis in technical systems and processes. Fault diagnosis is becoming one of the largest domains where expert systems are find application from their early stages. The process of diagnosing faulty conditions varies widely across to different approaches to systems diagnosis. The application of decision-making knowledge based methods to fault detection allows an in-depth diagnosis by simulating the human reasoning activity. Most of the past applications have been rule based while the automation of the diagnostic process including real-time data and/or modelling techniques added a new dimension to diagnostic task by detecting and predicting faults on line. Combination of expert systems technology with other artificial intelligent methods or with specific classical numerical methods adds more effectiveness to the diagnostic task. These knowledge based diagnostic techniques are presented in this Chapter, technical details of their implementation are provided, the advantages and drawbacks of every technique are outlined, examples from recent research work of expert diagnostic practice in industry are presented and current research trends are underlined to help the reader to delve into the matter.

INTRODUCTION

Fault diagnosis in technical systems has received a lot of theoretical and practical attention over the last years. Diagnosis is a complex reasoning activity, which is currently one of the domains where Artificial Intelligence techniques have been successfully applied as these techniques use association rules, reasoning and decision making processes as would the human brain in solving diagnostic problems.

[1] Department of Mathematics and Computer Science, Technological Education Institute of Piraeus, Konstantinoupoleos 38, N. Smirni, GR-171 21 Athens, Greece

A variety of fault detection and diagnosis techniques have been developed for the diagnostic problem solving process. These techniques include model based approaches, knowledge based approaches, qualitative simulation based approaches, neural network based approaches and classical multivariate statistical techniques.

Expert systems found broad application in fault diagnosis from their early stages because an expert system simulates human reasoning about a problem domain, performs reasoning over representations of human knowledge and solves problems using heuristic knowledge rather than precisely formulated relationships, in forms that reflect more accurately the nature of most human knowledge.

Strategies and capabilities for diagnostic expert systems have been evolving rapidly. Fault diagnosis for technical systems and processes need experiential knowledge in parallel to scientific knowledge for the effective solution of the diagnostic problem solving process. Different diagnostic approaches require different kinds of knowledge about the process. These approaches include first principal knowledge governing the process operation, empirical knowledge such as operators' experiences and historical data about the process operation under various normal and faulty conditions. From the early stage, when Feigenbaum (1981) published the reference for the early expert systems, numerous systems have been built in a variety of domains. Early diagnostic expert systems were rule-based and used empirical reasoning whereas new model-based expert systems use functional reasoning.

Automated diagnostic applications require diagnostic conclusions on-line under time constrains. These expert systems should be able to interpret signals as well as to deliver the required control action, conduct tests and recommend diagnostic procedures. For this purpose these systems should use a combination of quantitative and qualitative methods for fault detection that allows interaction and evaluation of all available information sources and knowledge about the technical process.

In this Chapter these strategies will be examined, the nature of automatic expert diagnostic and supervision systems will be revealed and recent trends in expert systems development will be described. In addition, a reference to the evolution of knowledge acquisition, knowledge representation techniques as well as user interface functions for expert systems will be provided. Examples from recent expert diagnostic practice in industry will be presented as well.

EVOLUTION OF EXPERT SYSTEMS TECHNOLOGY FOR FAULT DETECTION

In the late 1960's to early 1970's, expert systems began to emerge as a branch of Artificial Intelligence. The intellectual roots of expert systems can be found in the ambitions of Artificial Intelligence to develop "thinking computers". Domain specific knowledge was used as a basis for the development of the first intelligent systems in various domain. Feigenbaum (1981) published the best single reference for all the early systems. In the 1980's, expert systems emerged from the laboratories and developed commercial applications due to the powerful new software for expert systems development as well as the new possibilities of hardware.

Feigenbaum (1982) defined an expert system as "an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution". Differences from conventional programs include facts such as: An expert system simulates human reasoning about a problem domain as the main focus is the expert's problem solving abilities and how to perform relevant tasks, as the expert does. An expert system performs reasoning over representations of human knowledge in addition to doing numerical calculations or

data retrieval using the knowledge base and the inference engine separately. An expert system solves problems using heuristic knowledge rather than precisely formulated relationships in forms that reflect more accurately the nature of most human knowledge dealing with symbolic values and procedures.

The first diagnostic expert systems for technical fault diagnosis were developed in the early 1970's at MIT as is reported by Scherer and White (1989). Since then numerous systems have been built. Surveys of the first diagnostic expert systems of technological processes are provided by Pau (1986), Tzafestas (1989), Scherer and White (1989). During the period 1985-1995 expert systems were the "hot topics" in artificial intelligence developments due to the attention in knowledge management issue. After 1995 expert system applications started to decline as stand-alone systems and their technology embedded in mainstream of information technology. New expert systems started to combine symbolic with numerical information or with other artificial intelligence techniques to produce more effective systems.

From the early systems gradually emerged a dominant architecture based on separation of domain knowledge and general reasoning knowledge knowing as inference engine. Inference engine was relied on search techniques backward and forward changing. First generation of expert systems was rule-based while later on the model-base approach was emerged,

Rule-Based Diagnostic Expert Systems

Rule-based expert systems have a wide range of applications for diagnostic tasks where expertise and experience are available but deep understanding of the physical properties of the system is either unavailable or too costly to obtain.

In the rule-based systems, knowledge is represented in the form of production rules. A rule describes the action that should be taken if a symptom is observed. The empirical association between premises and conclusions in the knowledge base is their main characteristic. These associations describe cause-effect relationships to determine logical event chains that were used to represent the propagation of complex phenomena. The general architecture of these systems includes domain independent components such as the rule representation, the inference engine and the explanation system. Basic structure of a classical rule-based expert system is presented in Figure1.

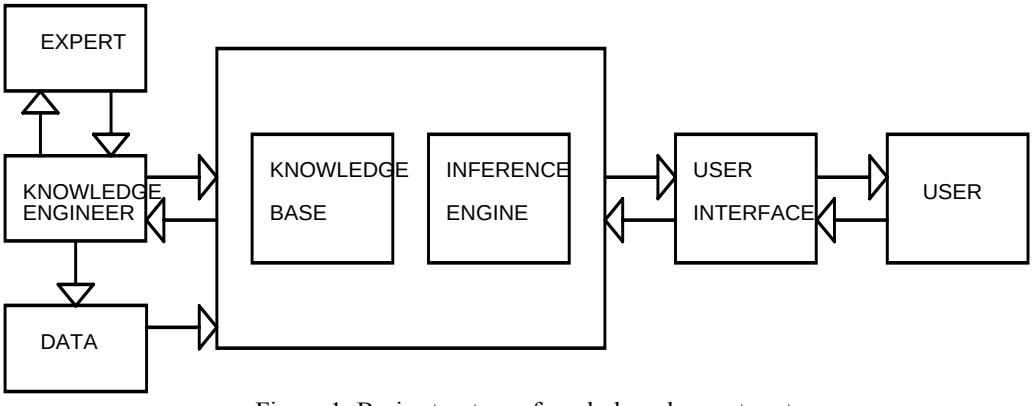


Figure 1: Basic structure of a rule-based expert system

Experiential knowledge suitably formatted consists the basis for the classical expert system approach. Fault diagnosis requires domain specific knowledge formatted in a suitable knowledge representation scheme and an appropriate interface for the human-computer dialogue. In this system the possible symptoms of faults are presented to the user in a screen where the user can click the specific symptom in order to start a searching process for the cause of the fault. Additional information about checking or measurements is used as input that, in combination with stored knowledge in the knowledge base guide to a conclusion.

Widman et al (1989) have counted the limitations of the early diagnostic expert systems as follows:

1. Inability to represent accurately time-varying and spatially varying phenomena.
2. Inability of the program to detect specific gaps in the knowledge base.
3. Difficulty for knowledge engineers to acquire knowledge from experts reliably.
4. Difficulty for knowledge engineers to ensure consistency in the knowledge base.
5. Inability of the program to learn from its errors.

The rule-based approach has a number of weaknesses such as lack of generality and poor handling of novel situations but it also offers efficiency and effectiveness for quasi-static systems operating within a fixed set of rules. In the modelling of human problem solving process it can guide users in a step by step manner to achieve a conclusion.

A decision tree is the main technique that defines the various logical paths that knowledge base must follow to reach conclusions. From the decision tree the relevant rules to each node can be written and so the initial knowledge base can be constructed.

Problems that are easily represented in the form of a decision tree are usually good candidates for a rule based approach. In the following example a rule is presented as it is needed to make a decision:

```
if      ?'reduced pressure' is Yes and
        ?'down' is No and
        ?'motor' is No
then    ?'electrical failure' is Yes.
```

In this rule the system searches for the topics 'reduced pressure', 'down' and 'motor' to satisfy the rule. Each of these rules may be further a set of rules, or simply a question asked to the user.

Rule based systems do not require a process model; however they do require a multitude of rules to cover all possible faults in a technical system and have difficulties with unexpected operations or new equipment. Among the main limitations of the early diagnostic expert systems are considered the inability to represent accurately time-varying and spatially varying phenomena, the inability of the program to learn from errors as well as the difficulties for knowledge engineers to acquire knowledge from experts reliably.

Most of the early expert systems were mainly laboratory prototypes that only briefly made into full operation in an industrial environment.

Model-based diagnostic expert systems

In model-based fault detection a model (mathematical or heuristic) is employed to describe the nominal behaviour of the monitored system. The generated residual signals that indicate differences between the model's output and measured process output are interpreted and evaluated to isolate faults.

Fault detection is realised after checking some measurable variables of a system in regard to a certain tolerance of the normal values and taking an appropriate action when a limit value is exceeded. Residuals are generated by comparing the sensed measurements to the predicted output of a model. The residuals are expected to be close to zero in fault free cases, but are distinguishably different from zero in the case of a fault in the system.

Model-based reasoning is a broad category that describes the use of variety of engineering models and techniques for fault diagnosis.

Model-based diagnostic expert systems have eliminated some limitations of the early expert systems. In these systems expert knowledge is contained primarily in a model of the expert domain. Model-based diagnosis uses knowledge about structure, function and behaviour and provides device-independent diagnostic procedures. The use of models enables the estimation of variables and parameters which are influenced by the fault. In addition model-based methods have the potential of early detection of slowly developing faults in complex processes.

Model-based diagnostic expert systems offer more robustness in diagnosis because they can deal with unexpected cases that are not covered by heuristic rules. In addition these systems are able to detect incipient faults in a technical process. The development of knowledge bases of such systems is less expensive because they do not require field experience for their building and are more flexible in the case of design changes. Model-based diagnostic systems offer flexibility and accuracy but they are also domain dependent. Real time implementation of model-based expert systems use enormous amount of computational capacity. The construction and updating of their knowledge bases are very demanding.

The use of models enables the estimation of variables and parameters which are influenced by the fault. In addition model-based methods have the potential of early detection of slowly developing faults in complex processes. Figure 2 presents a diagram of diagnosis process using system's models.

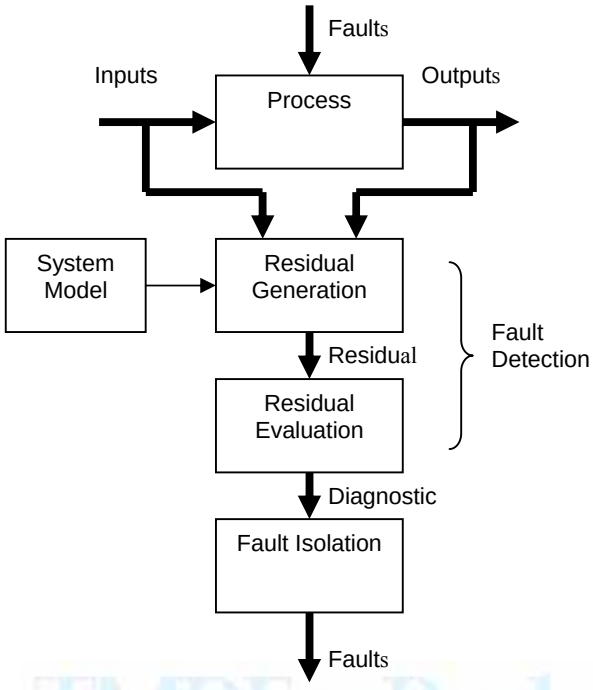


Figure 2: Fault diagnosis using system models

Difficulties with model based fault detection methods arise from the fact that the accuracy of the measurements needed to calculate the evolution of faults should be of high quality. In practice, fault detection systems make usually use of measurements from process instrumentation that is not necessarily installed for this purpose. In consequence, the instrumentation may not be sensitive enough and special sensors should be connected to the process equipment. Use of model-based methods may require assumptions about the process that are not valid, such as the assumption that the process is linear as well as that the influence of noise and disturbances to the fault detection process is of minor importance.

Recent contributions to model based fault diagnosis include Basseville and Nikiforov (1993), Patton et al (1995), Gertler (1998),Chen and Patton (1999), Soliman et al. (1998), Frank et al (2000), Cordier et al (2000), Leung and Romagnoli (2000), Mangoubi and Edelmayer (2000), Zhang et al (2002), Angeli and Chatzinikolaou (2002), De Kleer and Kurien (2003), Heim et al (2003), Korbicz et al (2004), Isermann (2005), Yusong et al (2006), Fekih et al (2006).

On-line diagnostic expert systems

On-line diagnostic expert systems usually use a combination of quantitative and qualitative methods for fault detection that allows interaction and evaluation of all available information sources and knowledge about the technical process. In these systems although basic diagnostic procedures are very satisfactory, real-time issue such as sensors drift can lead to problems with nuisance alarms in a system.

On-line use has also revealed the need for long and dead-time considerations along with simple high/low limit and rate detection.

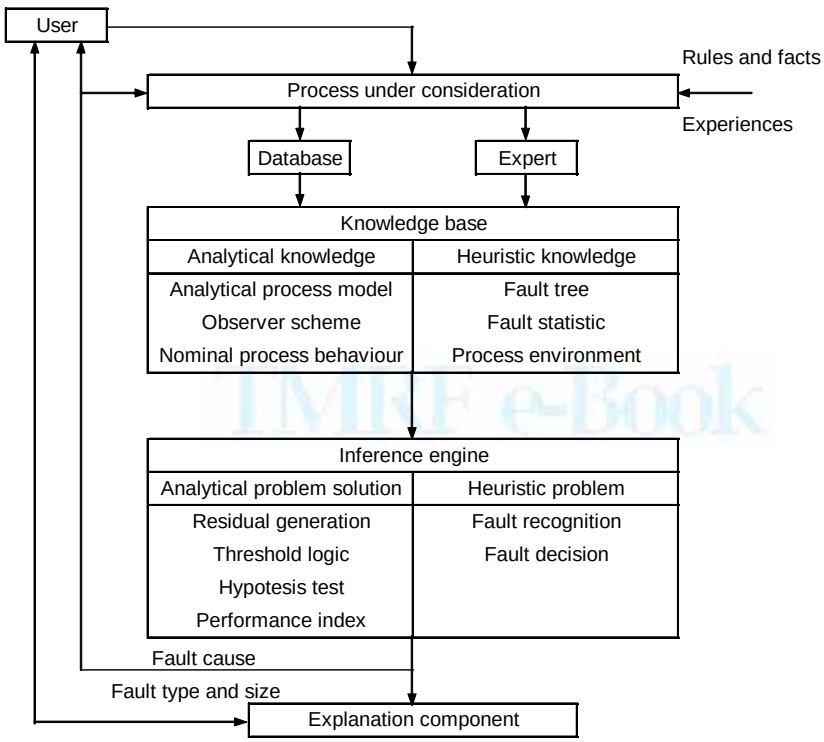


Figure 3: Basic structure of an on-line expert system (after Frank 1990)

The basic structure of an on-line expert system is presented in Figure 3. One of the main characteristics of this system is that in parallel to the knowledge base of the traditional expert system a data base exists with information about the present state of the process that interacts with the knowledge base of the system. The information of the data base is derived on-line from the sensors. This database is in a state of continuous change. The knowledge base of the system contains both analytical knowledge and heuristic knowledge about the process. The knowledge engineering task comprises different knowledge sources and structures. The inference engine combines heuristic reasoning with algorithmic operation in order to reach a specific conclusion. Response time is a critical issue for on-line expert systems because they are operating in parallel with a dynamic process.

Calculations and fault detection must be performed in a specific time in order to perform fault diagnosis and control in a suitable time. In addition, the quality of the incoming data from the external environment plays a primary role in the diagnostic process. Poor quality of data due to the time variation or sensor performance may be result inappropriate diagnostic conclusions.

When models of the technical systems are incorporated to the fault detection system using traditional, theoretical modelling techniques that are usually used for the detailed modelling of faults are not suitable for on-line performance of a system because these techniques need time for the running of the models of faults and in on-line systems it is required to reduce this time to the shortest possible. In addition, knowledge base development techniques of are not the required for on-line performance since updating from the sensor measurements and possibilities for the interaction of different sources of knowledge is needed.

In Table 1 the main advantages and disadvantages of each expert systems technology regarding the diagnostic processes for technical systems are summarized.

ADVANTAGES	DISADVANTAGES
Knowledge-based methods for fault detection	
a. Rule based diagnostic expert systems	
Rules can be added or removed easily	Lack of generality
Explanation of the reasoning process	Poor handling of novel situations
Induction and Deduction process is easy	Inability to represent time-varying and spatially varying phenomena
A process model is not required	Inability to learn from their errors
Efficiency and effectiveness in fault detection	Difficulties in acquiring knowledge from experts reliably
	Development and maintenance is costly
b. Model based diagnostic expert systems	
Device independent diagnosis	Domain dependant
Knowledge acquisition is not needed	Difficulties in isolation of faults
Ability of diagnosing incipient faults	Knowledge bases very demanding
Deal with unexpected cases	
Flexibility in the cases of design changes	
Dynamic fault detection	
c. On-line diagnostic expert systems	
Real time fault diagnosis	Domain dependant
Ability to handle noise	Good models are required
Generalization	Require considerable data
Fast computation	Inability to explain the reasoning process
Ability to handle with dynamics	Computationally expensive

Table 1: Expert system techniques for fault detection and diagnosis

The development environment

Historically the first expert systems were developed using the Lisp programming language as suitable for the manipulation of symbolic information. During 1980's Prolog gained some popularity. In the cased that speed and interaction with various kind of information was required their implementation language were considered. In 1990's logic programming languages were used for expert system applications. Expert system shells have been also used especially in 1980's providing a wide range of facilities for developers. In 1990's commercial products were extended with object oriented characteristics or real-time implementation features. Recently researchers have proposed interactive environments according to the needs of specific applications.

The development environment for expert systems applications in engineering systems should be suitable for interaction of several functional areas. On-line expert systems in engineering applications require, in addition to symbolic representations, high precision solutions that can be mathematically formulated.

The appropriate development environment should support capabilities of effectively coupling symbolic and numerical techniques. On the other hand expert system tools are not well-suited to dealing with numerical processing due to their focus on symbolic representations.

Recent research work of on-line expert systems

On-line knowledge driven techniques have been developed the last few years. A survey paper on the early attempts of applying on-line knowledge-based systems to solve problems in domains such as aerospace, communications, finance, medicine, process control and robotic systems is published by Laffey et al (1988).

The authors emphasise that the surveyed applications have not progressed beyond the prototype state. The main reason for this situation was the lack of suitable tools specifically built for real-time monitoring as well as the limitations of inference engines in high-performance to guarantee response times.

In following, recent published research work in diagnostic on-line expert systems for technical processes is presented in detail. These systems are selected after an extensive research through the literature. The presentation of these systems includes some details of their application while a more extensive presentation is out of the scope of this paper.

Angeli and Atherton (2001) developed an on-line expert system to detect faults in electro-hydraulic systems using on line connections with sensors, signal analysis methods, model-based strategies and deep reasoning techniques. Expert knowledge is contained primarily in a model of the expert domain. The final diagnostic conclusions are conducted after interaction among various sources of information.

Saludes et al (2003) reported a fault detection and isolation scheme for hydroelectric power stations based on a neural network and an expert system subsystems. The expert system stores knowledge acquired by operators for the diagnostic conclusions while the neural network has been trained with data automatically collected for one year in order to decide between normal and abnormal states. This system was in the implementation state at the time of the report.

Koscielny and Syfert (2003) presented main problems that appear in diagnostics of large scale processes in chemical, petrochemical, pharmaceutical and power industry and proposed an algorithm for decomposition of a diagnostic system, dynamical creation of fault isolation threads and multiple fault isolation, assuming single fault scenarios.

Yu Quian et al (2003) presented the development and implementation of an expert system for real time fault diagnosis in chemical processes that provides suggestion to the operator when abnormal situations occur. Industrial applications to the fluid catalytic cracking process in refinery are also presented.

Nabeshima et al (2003) reported an on-line expert system for nuclear power plants that uses a combination of neural networks and an expert system in order to monitor and diagnose the system status. The expert system uses the outputs of the neural networks generated from the measured plant signals as well as a priori knowledge base from the pressurized water reactor. The electric power coefficient is simultaneously monitored from the measured reactive and active power signals. Angeli and Chatzinikolaou (2004) have developed an on-line intelligent process monitoring and diagnosis system where acquired data from an actual electro-hydraulic system are recorded, analysed and presented in a suitable format to obtain the information needed for the control and detection of possible faults in proportional valves. The diagnostic system uses cooperatively information and knowledge for the final real-time diagnostic conclusions.

Carrasco et al (2004) reported an on line diagnostic system for the determination of acidification states on an anaerobic wastewater treatment plant that uses expert knowledge to determine the acidification state of the process and on-line measurements of system variables classified in linguistic information using fuzzy-based rules. The reasoning process is realised by evaluating the inference system for the given inputs.

Yusong Pung et al (2006) have described the architecture of an expert system for fault diagnosis in a hydraulic brake system where the acquired knowledge for the knowledge base is based on software simulation. This simulation-based knowledge generation in combination with fuzzy knowledge representation is used for the final diagnostic reasoning.

Current trends in diagnostic systems development

Several researchers combine numerical with qualitative methods the last few years and various methodologies have been proposed for the combination of knowledge-based techniques with numerical techniques considering that the combination of both approaches in an effective way offers an appropriate solution for most situations. In these knowledge-driven techniques, although the governing elements are symbolic, numeric computations still play an important role by providing certain kinds of information for making decisions.

Relevant recent research work is reported by Chen and Patton (1999), Frank et al (2000), Patton et al (2000), Manders and Biswas (2003), Nyberg and Krysander (2003), Saludes et al (2003), Koscielny and Syfert (2003), Persin and Tovornik (2003), Gentil et al (2004), Korbicz et al (2004), Wang (2006).

Current trends include coupling of these diagnostic techniques in order to produce more effective tools as well as combining expert systems technology with other artificial intelligence technologies as neural networks or genetic algorithms to cover various requirements of the diagnostic process. Hybrid

intelligent systems can offer powerful tools to complex problems (Patton et al 2000; De Kleer 2003; Nyberg et al 2003; Biswas 2004; Liu et al 2005; Su and Chong 2006; De Boer and Klingenberg 2008).

Recently diagnostic expert systems are find application in network conditions. Web-based expert systems are based on traditional expert systems technology, rule-based and case based reasoning primarily. They incorporate client-server architectures and web browser-based interfaces (Grove 2000; Duan et al 2005). Fault diagnosis in networked condition offer new possibilities for the diagnostic task (Fang et al 2006).

EVOLUTION OF KNOWLEDGE AQUISITION TECHNIQUES

The knowledge acquisition bottleneck has not been solved but over past years research in that area has provided some useful results. In the field of expert systems, the distinction between "deep or model-based or fundamental" systems and "shallow or compiled or empirical" systems has been given considerable attention by researchers over the last years. Limitations of knowledge acquisition process for technical and anthropological reasons result in limitations of acquired empirical knowledge. In addition, the specific problem dependent nature of the empirical knowledge lacks in robustness compared to the deep models. On the other hand, scientific knowledge of the model-based systems cannot cover the total range of diagnostic tasks, since the diagnostic activity is based mainly on experience.

Building an expert system involves the elicitation of the expertise of a human expert and the translation of the knowledge thus attained into a machine-executable form. Knowledge acquisition is classically referred to as the bottleneck in expert system development by Feigenbaum (1981), since the resulting expert system depends on the quality of the underlying representation of expert knowledge. Unfortunately, an adequate theoretical basis for knowledge acquisition that requires a classification of knowledge domains and problem-solving tasks as well as an improved understanding of the relationship between knowledge structures in human and machine has not been established (Foley and Hart 1992).

Experiential knowledge is acquired over a number of years' experience and consists of much more than just facts and rules. If all symptoms for all malfunctions are known in advance then a simple table lookup would be adequate for the fault specification. The usage of the expertise, that involves weighing up different pieces of evidence in order to choose the most appropriate one for the specific situation using sometimes unknown heuristics, helps decidedly in the problem solving process and particularly in making useful or accurate decisions or diagnoses.

Over the last few years, researchers from the field of cognitive science have proposed a variety of techniques for the knowledge acquisition process. These techniques range from the formal to informal, from those which are driven by the knowledge engineer to those which are driven by the expert and from those which are visible to the expert to those which are hidden. Interviews, protocol analysis and repertory grid analysis are characterised as traditional techniques to the knowledge acquisition process. Automatic induction techniques have also been proposed by researchers. Interviews are considered as the most popular and widely used form of expert knowledge acquisition (Hoffmann 1992; Foley and Hart 1992). One serious drawback of interviews is the fact that they are time-consuming. In addition, the effectiveness of interviews depends on the skill of the interviewer in posing suitable questions and the skill of the expert in conveying his knowledge and techniques.

Task performance and protocols hand the task of guiding the "interview" over to the expert by requiring that the expert thinks-aloud while working through a series of either simulated or written case examples. In this method, the study of an expert's action is sometimes called protocol analysis. It has been adopted from the cognitive science and has been automated quite early in artificial intelligence (Waterman and Newell 1971). This method is considered by Holsapple et al (1994) as more efficient for complex cases than for relatively simple cases.

Repertory grid method yields a set of dimensions defining the space which contains the domain objects. The knowledge engineer decides on a list of elements first and then presents them in trees asking the expert to indicate how one of them differs from the other. This technique is useful in clustering information but requires a lot of time to work effectively.

Researchers try to automate the knowledge elicitation process from early stages by proposing knowledge acquisition software methods. Attempts in this direction had produced a great variety of approaches such as Gaines and Boose (1988), Michie (1982) and a great number of implemented systems such as by Boose et al (1987), Quinlan (1986), Clement (1992). Knowledge acquisition software methods include machine induction or induction by example tools and knowledge elicitation tools.

Mettrey (1992) considers induction tools as ineffective for complex applications where knowledge representation and reasoning capabilities are required. Knowledge elicitation tools consist of computer programs that guide the expert to enter information about a domain into a computer and to classify this information in order to generate rules from the organised data. Tools for automatic knowledge acquisition are described by Newquist (1988), Diederich et al (1987), Clement (1992), Badiru (1992).

The next decade, new arrivals such as neural networks (Cooke 1992), genetic algorithms (Odetayo 1995) and hypertext (Pouliezos and Stavrakakis 1994) have also been proposed as automated knowledge acquisition methods.

Cooke (1992), criticising the common knowledge elicitation techniques, considers, firstly, that they have limitations to the extent that they first rely on verbal knowledge which is often inaccurate and incomplete; secondly that they miss much knowledge which is automatic or compiled; thirdly that they produce output that requires extensive interpretation in order to transform it into a computer usable format; and fourthly that they ignore the organisation or the structure of the facts and rules elicited which is a critical point because experts differ to novices not only in the facts and rules they use, but also in the way that the facts and rules are organised in their memories.

Automated methods have also been criticised in that most of the proposed knowledge engineering tools are used to translate the already elicited knowledge into a computer program rather than to perform knowledge elicitation itself.

Much recent research effort in the field of knowledge acquisition (KA) has focused on extending knowledge acquisition techniques and processes to include a wider array of participants and knowledge sources in a variety of knowledge acquisition scenarios as (Xiong et al 2002; Wagner et al 2002; Xing et al 2003; Wagner et al 2003; Wu et al 2003; Gale 2005; Chen et al 2008). As the domain of expert systems applications and research has expanded, techniques have been developed to acquire and incorporate knowledge from groups of experts and from various sources.

One of the advantages of the *model-based* expert systems is the avoidance of the knowledge acquisition process as the knowledge is involved in the embedded model of the domain. But on the other hand, model-based diagnostic systems are criticised as not always being able to pinpoint the faulty component (Koseki 1992), and sometimes a lot of tests are required to reach a conclusive decision due to the lack of heuristic knowledge. Hart A. (1992) has also pointed out that no diagnostic analysis is complete without face-to-face discussion with the expert.

In on-line expert systems the knowledge engineering task is different since the knowledge engineer has to combine different knowledge sources and structures. The technological or scientific knowledge (knowledge of functioning) from the domain model has to be combined with the experiential knowledge (knowledge of utilisation) from the domain expert. The characteristics of knowledge to be acquired depend on the nature of the domain as well as on the level of expertise of the domain expert. Current trends include experiential knowledge to complement the scientific knowledge in order to model more precisely the expert's reasoning activity, and to gain the efficiency of heuristics and the advantages of a real world application.

The empirical knowledge and the scientific knowledge usually solve different parts of the overall problem domain cooperatively. Deep knowledge involves concepts of cause that are not available to the relatively compiled knowledge. Empirical knowledge is particularly useful in the diagnostic phase since the target is to find the specific faulty element and not only to declare a faulty behaviour of the system but to propose specific actions. Scientific knowledge is useful for representing the dynamic behaviour of a system or for predicting future faults. The interaction of the two types of knowledge driven by the current problem solving circumstances gives a power to the interaction process.

Experiential knowledge is acquired by the domain expert for specific problem solving requirements. This knowledge refers to the limits for detecting faults, to the decision about a faulty element in specific cases, to the detection of multiple faults as well as to giving advises on the way the logic signal information should be used effectively so that to avoid additional checks in detecting faults. There is no universal agreement on which knowledge acquisition technique should be used in a specific case. The knowledge engineer should decide on the suitable knowledge acquisition method by weighing up the ease with which they can be used against the quality and amount of information being obtained, according to Hart (1992). The choice of the knowledge acquisition method depends on problem domain factors such as size and complexity. It is common to start with interviews and then apply other methods when considered useful.

Knowledge acquisition phase could conclude using diagrams to clarify that the acquired knowledge was suitable for solving the problem. The data organised in diagrams specify which data item is directly or indirectly related to others and in what order. At the end the expert criticise the diagrams, provide missing information and corrected any errors in this formulation. These diagrams led easily to the final construction of the decision tree in the knowledge representation phase.

EVOLUTION OF KNOWLEDGE REPRESENTATION TECHNIQUES

The available knowledge must be properly represented for the problem solving procedures. In the development of knowledge-based systems, an essential element lies in the organisation of the knowledge and the reasoning strategies. Various knowledge representation techniques are available. Some of them are suitable for the majority of problems. Basic data-structures generally used for symbolic representations are production rules, semantic networks, frames, predicate logic and hybrid

architectures. The choice of the most suitable representational schema depends on the type of procedural control required and the degree of familiarity of the knowledge engineer with a technique.

There are, however, problems that require unique knowledge representation techniques. The engineering diagnosis process is typically a mixture of empirical and functional reasoning coupled with hierarchical knowledge. Formalism is needed to hierarchically express the structural behavioural knowledge.

In *rule based expert systems* the rules corresponding to the nodes of the decision tree can be written. In *model-based expert systems* knowledge could represent by the mathematical model of the system. In an *on-line expert systems* the knowledge engineering task requires a knowledge representation model suitable for the integration and combination of the two different knowledge sources and structures that are involved in the problem solving process: the first principal knowledge or scientific knowledge and the empirical knowledge.

Current trends include representation of different nature of knowledge can be of different types of representation model. Their interaction should not require that both types of knowledge are of the same representation, or the problem-solving methods that use this knowledge are the same.

The scientific knowledge could represent by the mathematical model of the system in a numerical formation and the experiential by the knowledge base of the system in a symbolic formation. Scientific on-line knowledge coming from sensor measurements should interacted with both the knowledge of the mathematical model and the knowledge base of the system. The representation and the on-line interaction of all these types of knowledge initially required a suitable environment.

EVOLUTION OF USER INTERFACE TECHNIQUES FOR EXPERT SYSTEMS

The User Interface, the third major component of an expert system which allows bi-directional communication between system and user is considered to be a critical part of the success of an expert system.

It has been argued that user interfaces of expert systems are even more troublesome and vital than those of traditional systems because the expert system must present not only the conclusions of the task but also the explanation of the processes by which the conclusions are reached. Consequently the system does not just assist with task performance, but actually assists with decision making about tasks and task performance (McGraw 1992).

The two main areas in the construction of expert systems that involve interface issues are the type of dialogue control and the explanation facilities. In on-line expert systems the role of the user interface is different because they operate in relation to a continuous process and the response time is a critical issue. In addition, on-line systems operate autonomously so that the role of the dialogue control is quite restricted. According to Buck (1989), these systems do not generally respond to user-initiated interrogations but to the process-initiated situation, and in this respect they differ from traditional advisory systems as well as from controller systems that only transfer information from the system to the controlled process. Thus the user cannot directly control their behaviour.

Even on-line systems that are operating under time constraints must be able to explain the reasoning process at the user's request or to give advice quickly to the user. The advice can be produced at any time while the system is executing different procedures. The explanation of the reasoning can be produced in a specific time after entering a required piece of information. This is not acceptable when the system is controlling or monitoring a dynamically changing process.

A good interface should serve the user in a comprehensible way that matches the manner that a user conceptualises the task in terms of syntactic requirements and semantic organisation and the performance of the task since, from the user's point of view, the interface is the system. Intelligent systems offer new opportunities for more flexible dialogue.

A key aspect of developing a useful interface is the notion of conceptual models designed into interfaces by their designers and the actual mental models that are developed in the user's mind as they interact with the systems.

The determination of the user's views to the domain and to the system has as a consequence the experimentation with ways to transfer this understanding to user interface components by selecting a suitable metaphor. People usually attempt to make sense of something new by comparing it with and generalising from what they know about something similar (metaphorical reasoning). A successful metaphor allows operations that are similar to those in the user's problem domain. The problem is to find metaphors with familiar or easily interpreted objects and actions within a system.

Researchers have proposed various models for human-computer interaction analysis. Card, Moran and Newell (1980) proposed the GOMS (Goals Operators Methods and Selection rules) to support the explicit analysis of user tasks at different levels of description. This framework is intended to evaluate the usability of the interface design.

Norman (1986) postulates that the goal of the user and the system state differ significantly in form and content so that it creates a gulf between them. The problem is related to the difficulties of linking psychological goals to the physical variables and controls of the task. This gulf model includes the terms "gulf of execution" and the "gulf of evaluation" between users' goals and knowledge and the level of description provided by the system. It can be bridged using commands and mechanisms of the system to match the user's goal as much as possible.

Shneiderman's (1992) syntactic / semantic model highlights that users have syntactic knowledge about device-dependent details and semantic knowledge about the concepts. Semantic knowledge has two components; task domain and computer domain. Within the domains knowledge is divided into actions and objects. This knowledge is concerned with the meaning of objects and deals with relationships between objects and their attributes. Syntactic knowledge is arbitrary and therefore acquired by memorisation.

Many researchers have criticised the models that deal with only one of the gulfs presented by Norman (1986), the gulf of execution (how we convert our intention into action) and don't deal with the gulf of evaluation (how we interpret the system state) or that models are too complex to be used and to understand on an everyday basis.

Many of the modelling techniques offer the designer a way to analyse a system without directly involving the user. The advantage of this approach is that involving the user is often time consuming.

The disadvantage is that modelling techniques only inform us of the potential difficulties that users might encounter due to the lack of information of the real problems that users experience. It is often pointed out that developments in the area of user models will lead to the provision of better interface functions. Very few expert systems being developed at present entail more than the very simplest of user models as much work in this area is at the research level.

Concepts of the models can provide usefulness in systematising efforts for designing interactive systems or in comparing different possible designs or in telling in advance exactly how a system should be designed from a human point of view. The end users of this system are mainly engineers. They prefer more schematic diagrams to the problem solving strategies they use. A style of interface with graphical explanations, hypertext or windowing is needed.

The explanation facility, one of the key characteristics that sets expert systems apart from traditional software systems, improves the ability of the system to totally mimic the human expert as it can provide explanations for the basis of its conclusions. During the testing of the development process of an expert system, the explanation facility helps developers to test and debug the system. A deep explanation facility enables users to check whether the system is taking important variables and knowledge into account during the reasoning process.

User interfaces for expert systems that adapt to the users are beginning to be developed. This includes adapting screen messages, help facilities and advice to meet the user's needs. The user model is the main source that makes explicit assumption about the user useful for the design of system responses, help or explanations which meet the expected needs of the user.

AN EXPERT SYSTEM MODEL FOR ON-LINE FAULT DIAGNOSIS

In following the reasoning process of a model expert system for on-line fault diagnosis will be presented. The reasoning process for real-time dynamic systems requires on-line interaction of various sources of available information in order to produce a reliable and useful knowledge based diagnostic system. In this example an effective interaction of real-time numerical information obtained by an actual system and symbolic knowledge for diagnostic purposes. The decision making process is performed after co-operation of dynamic modelling information, on-line sensor information, and stored knowledge using suitably formatted DASYLab modules.

This expert system example detects faults in hydraulic systems after suitable interaction of knowledge and information. This process is performed on-line and the system is able to respond to dynamically changing states by combining modelling information, on-line sensor measurements and symbolic data.

The actual system used for the experimentation process of this work was a typical electro-hydraulic system consisting of a hydraulic motor controlled by a proportional 4-way valve with a cyclical routine which requires a high speed for a short period of time and then returns to a low speed.

The actual system was modelled in terms of mathematical equations and the simulation program was written in C programming language. The developed mathematical model takes into account the non-linear character of hydraulic systems and the incompressibility of the hydraulic fluid in the pipes as well as the special characteristics of the hydraulic elements used. The technical specifications and function curves were used in order to obtain parameter values given by the manufacturer, after laboratory testing, and the quasi-steady character of the hydraulic elements is also taken into account

in order to reduce the complexity of the equations, so that the model can describe more accurately the actual system.

The DASYLab software was used for data acquisition and control purposes as well as for comparison of measured values and relatively calculated quantities using suitably connected modules. The expert system is developed in two parts. The first part performs the numerical calculations while the second part performs the symbolic representations. A data acquisition, monitoring and control module generates and interprets signal values coming from an actual hydraulic system. Measurable quantities of the variables correspond to the pressure at critical points of the hydraulic system and the angular velocity as well as the digital input signals are transferred to the expert system for the decision making process.

The mathematical model of the actual system that generates additional information for comparison with measured variables is also involved in the fault detection method. The evaluation of deviation limits between the performance of the system and the performance of the fault free model stimulates the fault detection procedure. The diagnosis of faults is performed through the knowledge base of the expert system. This system requires a knowledge base architecture that permits the interaction of sensor information, modelling information and experiential knowledge symbolic representation.

The fault diagnosis process

The measured values of the important system variables are compared with the calculated quantities from the simulation program and their deviation is translated to qualitative information. Measured and calculated values of the important system variables are compared in the first part of the expert system that has been developed using the capabilities of the DASYLab software. In following the on-line comparison process between measured and simulated values of the pressures p_a, p_b and the angular velocity ω is presented as well as the transformation of the comparison results to linguistic variables. These linguistic values are used for the fault detection process and the final diagnostic conclusions by the knowledge base of the expert system. The comparison process of measured and calculated values of pressure at different point and the angular velocity are presented in Figure 4.

Figure 4 represents a part of the DASYLab “worksheet” developed for that purpose. The module “outmo3:” corresponds to a file “outmo3.asc” that contains the numerical data from the simulation process. These values are read in order to be compared with the measured values. The “Action00:” module is used to initiate various actions. The “outpr3:” module is used to read the measured values of the pressures p_a, p_b from the file "outpr3" that is produced by the data acquisition system.

The processing of the pressure p_a is performed by the following modules: The “Y/t Chart00:chart” module displays the measured values of the pressure p_a. The " Pam-Pas:" module calculates the differences between the pressure and the measured values of the pressure p_a. The "abs(DPa):" module calculates the absolute values of the difference of pressure p_a , Pam-Pas. The " Statistics03:" module calculates the mean value of the differences between Pam-Pas. The " Pa:" module sends the message "OK" in the case that the difference between Pam and Pas is 0 to3 bar, "SMALL" in the case that this difference is 3 to 6 bar and "MED" in the case of difference 6-10 bar to the message module “DPo OK?”. This information is used by the expert system to determine more precisely a faulty element. This “DPa OK?:” is a message module that writes the values "OK", "SMALL" and "MED" to the text file "fpa.txt". Similar processing has been performed for the pressure at other points of the hydraulic system.

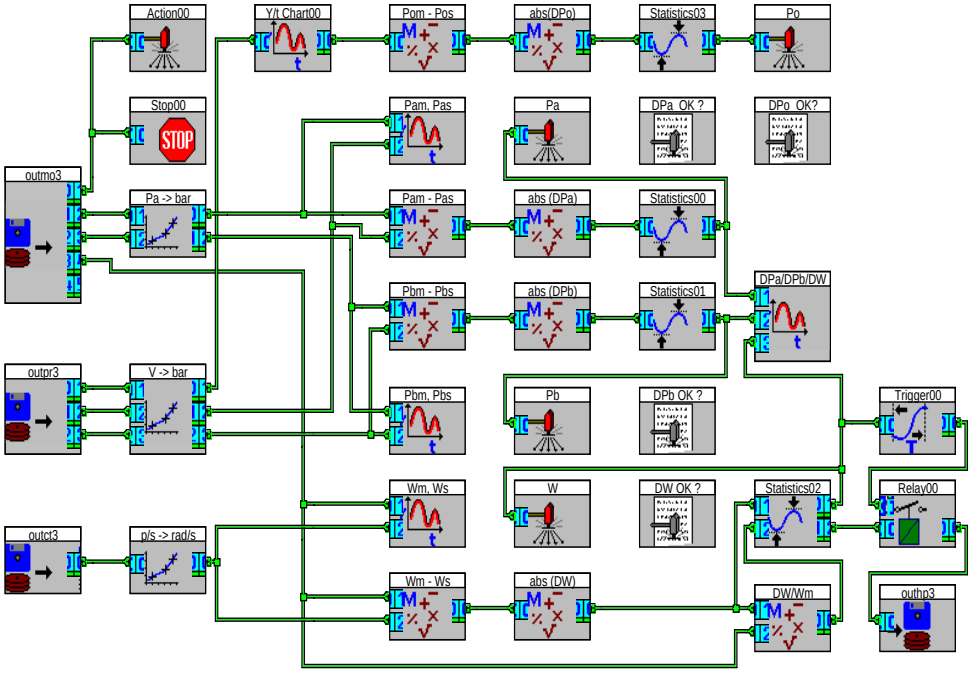


Figure 4: Comparison of measured and calculated data

Output of this worksheet are the files "fpa.txt", "fpb.txt" etc. that include text information about the presence of a fault in p_a, p_b, respectively. These files are used by the expert system as input together with the text file information coming from the data acquisition process.

This symbolic information can be passed in the structure of the knowledge representation scheme and can trigger specific set of rules that are organised under the structure of the topic.

The knowledge base development

The interaction of the various sources of information and knowledge was realised by knowledge representation scheme the "topic". This programming structure offers the opportunity to read external linguistic information from files that can be combined with the stored knowledge. Rules are embedded in topics so that the structure of the final application is a collection of topics. Rules that refer to general assumptions and are represented to specific branches of the decision tree are grouped and embedded in a specific topic. In the structure of a "topic" interact stored knowledge in rules and external information from files coming directly from the data acquisition system pre-processed and transformed to linguistic values.

The interaction of all available information sources in the structure of a "topic" is schematily presented in Figure 5. The symbolic representation of the empirical knowledge and the part of scientific knowledge embedded on the circuit diagrams is realised using the second part of the expert system while the first part is used for the representation of the scientific knowledge coming on-line from sensors, the results of the performance of the mathematical model and the comparison of the results of

both interactions. These comparison results continuously update the knowledge base of the expert system.

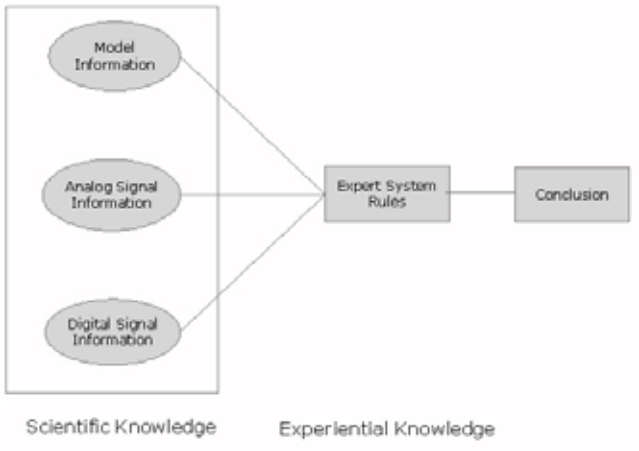


Figure 5: The knowledge base system architecture

SUMMARY AND CONCLUSION

Diagnostic problems are considered as ill-structured problems where there are no efficient algorithmic solutions because all the symptoms for all faults are not known in advance. The effectiveness of diagnostic reasoning lies in the ability to infer using a variety of information and knowledge sources, connecting or selecting between different structures to reach the appropriate conclusions.

Expert systems technology has been widely adopted by many software development companies and industry. Although originally expert system were seen as stand- alone systems now are components of large information technology architectures. As applications become more complex for the requirements of increasing automation, the suitable technique to perform diagnostic reasoning can be quite challenging. The challenge of the future is the development of generic diagnostic architectures that can potentially use a variety of diagnostic techniques, process independent and modular in design so that they can be applied to all the essential equipment.

In this paper, the evolution of expert systems paradigm for the solutions of the diagnostic problem have been presented as well as the evolution of knowledge acquisition, representation and user interface methods for the expert systems development process. Experiential knowledge, scientific knowledge and a combination of the two sources of knowledge has been used to perform the diagnostic task.

Particular emphasis has been posed on the on-line expert system paradigm for technical systems. In this case the main characteristics of the technology as well as a detailed description of specific expert system applications of the last years for the on-line fault detection process have been illustrated. In addition, difficulties and main problems of each technology regarding the diagnostic process for technical systems have been discussed and future directions of the research in these systems have been also highlighted.

Current trends include coupling of these diagnostic techniques in order to produce more effective tools. The new trend for the realization of fault diagnosis is the implementation with cooperative functions that are also distributed in network architecture.

In conclusion, this paper emphasises in expert systems development for fault diagnosis in technical processes. The main characteristics of each technology as well as the difficulties and the problems arising in expert systems development have been underlined and specific expert system paradigms have been presented. Future directions of the research in these systems have been also highlighted.

REFERENCES

Angeli, C. and Atherton, D.P. 2001. A Model Based Method for an on-line Diagnostic Knowledge-Based System. *Expert Systems,* 18(3):150-158.

Angeli, C. and Chatzinikolaou, A. 2002. Fault Prediction and Diagnosis in Hydraulic Systems. *Journal of Engineering Manufacture*, 216(2):293-297.

Angeli, C. and Chatzinikolaou, A. 2004. On-Line Fault Detection Techniques for Technical Systems: A Survey. *International Journal of Computer Science & Applications,* 1(1):12-30.

Badiru, A. 1992. *Expert Systems Applications in Engineering and Manufacturing:* Prentice-Hall Inc. N. Jersey.

Basseville, M. and Nikiforov, I. 1993. *Detection of Abrupt Changes: Theory and Application*: Prentice Hall.

Biswas, G., Cordier, M., Lunze, J., Trave-Messuyes, L. and Staroswiecki, M. 2004. Diagnosis of Complex Systems: Bridging the Methodologies of the FDI and DX communities. *IEEE Transactions on Systems, Man and Cybernetics*, 34(5).

Boose, J. H. and Bradshaw, J. M. 1987. Expertise Transfer and Complex Problems using AQUINAS as a Knowledge-Acquisition Workbench for Knowledge-Based Systems. *International Journal of Man-Machine Studies,* 26:3-28.

Buck, L. 1989. Human Operators and Real-Time Expert Systems, *Expert Systems*, 6(4):227-237.

Card, S., Moran, T. and Newell, A. 1980. Computer Text-Editing: An Information-Processing Analysis of a Routine Cognitive Skill. *Cognitive Psychology*, 12:32-74.

Carrasco, E., Rodriguez, J., Punal, A., Roca, E. and Lema, J. 2004. Diagnosis of Acidification States in an Anaerobic Wastewater Treatment Plant using a Fuzzy-Based Expert System. *Control Engineering Practice*, 12(1):59-64.

Chen, J. and Patton, R. 1999. Robust Model Based Fault Diagnosis for Dynamic Systems: Kluwer Academic Publishers.

Chen C-H, Zhiming Rao 2008. MRM: A Matrix Representation and Mapping Approach for Knowledge Acquisition, *Knowledge-Based Systems*, 21(4):284-293.

Clement, P. R. 1992. Learning Expert Systems by Being Corrected. *International Journal of Man-Machine Studies,* 36:617-637.

Cooke, N. 1992. Eliciting Semantic Relations for Empirically Derived Networks. *International Journal of Man-Machine Studies,* 37:721-730

Cordier, M., Dague, P., Dumas, M., Levy, F., Montmain, J., Staroswiecki, M. and Trave-Massuyes, L. 2000. AI and Automatic Control Approaches to Model-Based Diagnosis: Links and Underlying Hypothesis. Proceedings *4th IFAC Symposium on Fault detection, Supervision and Safety of Technical Processes*, Budapest, 274-279.

De Boer T. and Klingenberg, W. 2008. Architecture for a Neural Expert System for Condition-Based Maintenance of Blankings. *International Journal of Materials and Product Technology*, 32(4):447-459.

De Kleer, J. and Kurien, J. 2003. Fundamentals of Model-Based Diagnosis, In Proceedings *Safeprocess 03*, Washington, U.S.A. 25- 36.

Diederich, J., Ruhmann, I. and May, M. 1987. KRITON: A Knowledge-Acquisition Tool for Expert Systems. *International Journal of Man-Machine Studies,* 26:29-40.

Duan, Y., Edwards, J. and Xu, M. 2005. Web-Based Expert Systems: Benefits and Challenges. *Information and Management*, 42:79-81.

Eriksson, H. 1992. A Survey of Knowledge Acquisition Techniques and Tools and their Relationship to Software Engineering. *Journal of Systems and Software*, 19(1):97-107.

Fang, H., Ye, H. and Zhong, M. 2006. Fault Diagnosis of Networked Control Systems. In Proceedings of the *SAFEPROCESS 06*, Beijing, 1-11.

Fekih A., Xu, H. and Chowdhury, F. 2006. Two Neural Net-Learning Methods for Model Based Fault Detection. In Proceedings of the *SAFEPROCESS 06*, Beijing, 72-83.

Feigenbaum, E. 1981. *Handbook of Artificial Intelligence:* Heuris Tech Press, W. Kaufman Inc.

Feigenbaum, E. 1982. *Knowledge Engineering in 1980's:* Department of Computer Science, Stanford University, Stanford CA.

Foley, M. and Hart, A. 1992. *Expert-Novice Differences and Knowledge Elicitation*: Springer Verlag, New York, USA.

Frank, P. 1990. Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy - A Survey and Some New Results. *Automatica*, 26(3):459-474.

Frank, P.M., Ding, X. and Koppen-Seliger, B. 2000. Current Developments in the Theory of FDI. In Proceedings of *IFAC SAFEPROCESS 2000*, Budapest, Hungary, 16-27.

Frank, P.M., Ding, X. and Marcu, T. 2000. Model-Based Fault Diagnosis in Technical Processes. *Transactions of the Institute of Measurement and Control*, 22:57-101.

Gaines, B. R. and Boose, J. H. 1988. *Knowledge Acquisition for Knowledge-Based Systems*. London: Academic Press.

Gale, W. 2005. A Statistical Approach to Knowledge Acquisition for Expert Systems. *Annals of Mathematics and Artificial Intelligence*, 2(1-4):149-163.

Galland, S. 1993. Neural Network Learning and Expert Systems. Cambridge, MA:MIT press

Gentil, S., Montain, J. and Combastel, C. 2004. Combining FDI and AI Approaches within Causal Model-Based Diagnosis, *IEEE Transactions on Systems, Man and Cybernetics*, 34(5):2207-2221.

Gertler, J. 1998. Fault Detection and Diagnosis in Engineering Systems. New York, Marcel Dekker.

Grove, R. 2000. Internet-Based Expert Systems. *Expert Systems*, 17(3):129-135.

Heim, B., Gentil, S., Celse, B., Cauvin, S. and Trave-Massuyes, L. 2003. FCC Diagnosis Using Several Causal and Knowledge Based Models. In Proceedings of SAFEPROCESS 03, Washington, U.S.A., 663-669.

Hoffman, R. 1992. The Psychology of Expertise: Cognitive Research and Empirical A.I., Springer Verlag, New York, USA.

Holsapple, C. and Raj, V. 1994. An exploratory study of two KA methods, *Expert Systems*, 11(2).

Isermann, R. 2005. Model-based Fault Detection and Diagnosis: Status and Applications. *Annual Reviews in Control*, 29:71-85.

Jiang, H. 2008. Study on Knowledge Acquisition Techniques, In the Proceedings of *2nd International Symposium on Intelligent Information Technology Applications*, 181-185.

Koscielny, J. and Syfert, M. 2003. Fuzzy Logic Application to Diagnostics of Industrial Processes. In Proceedings of SAFEPROCESS 03, Washington, U.S.A., 711-717.

Koseki, Y. 1992. Experience Learning in Model-Based Diagnostic Systems. In Proceedings of *Tenth National Conference on Artificial Intelligence*. July 12-16. AAAI Press/The MIT Press.

Korbicz, J., Koscielny, J., Kowalczuk, Z. and Cholewa, W. 2004., *Fault Diagnosis: Models, Artificial Intelligence, Applications*, Springer Verlag, Berlin.

Laffey, T., Cox, P. Schmidt, J., Kao, S. and Read, J. 1988. Real-Time Knowledge-Based Systems, *AI Magazine*, 9(1):27-45.

Leung, D., and Romagnoli, J. 2000. Dynamic Probabilistic Model-Based Expert System for Fault Diagnosis, *Computers and Chemical Engineering*, 24(11):2473-2492.

Liu, H., Chen, Y. and Ye, H. 2005. A Combinative Method for Fault Detection of Networked Control Systems. In Proceedings of 20th IAR/ACD Annual Meeting, 16-18 November, France, 59-63.

Manders, E. and Biswas, G. 2003. FDI of Abrupt Faults with Combined Statistical Detection and Estimation and Qualitative Fault Isolation. In Proceedings of *SAFEPROCESS 03*, Washington, U.S.A., 339-344

Mangoubi, R.S. and Edelmayer, A. M. 2000. Model Based Fault Detection: The Optimal Past, the Robust Present and a Few Thoughts on the Future. In Proceedings of *SAFEPROCESS 00*, Budapest.

McGraw, K. 1992. *Designing and Evaluating User Interfaces for Knowledge-Based Systems*: Ellis Horwood Limited.

Medsker, L.R. 1995. *Hybrid Intelligent Systems*. Kluwer Academic Publishers, Boston.

Michie, D. 1982. *The State of the Art in Machine Learning*. Gordon & Breach, London.

Milton, N. 2007. Knowledge Acquisition in Practice: A Step by Step Guide, Springer Verlag.

Nabeshima, K., Suzudo, T., Seker, S., Ayaz, E., Barutcu, B., Turkcan, E., Ohno, T. and Kudo, K. 2003. On-line Neuro Expert Monitoring System for Borssele Nuclear Power Plant, *Progress in Nuclear Energy*, 43(1-4):397-404.

Newquist, H.P. 1988. Braining the Expert. *Machine Studies,* 36:617-637.

Norman, A. 1986. Cognitive Engineering. In A. Norman & W. Draper, *User Centred System Design: New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates Inc.

Nyberg, M. and Krysander, M. 2003. Combining AI, FDI and Statistical Hypothesis-Testing in a Framework for Diagnosis. In Proceedings of *SAFEPROCESS 03*, Washington, U.S.A., 813-818.

Odetayo, O. M. 1995. Knowledge Acquisition and Adaptation: A Genetic Approach. *Expert Systems*, 12(1).

Patton, R., Chen, J. and Nielsen, S. 1995. Model-Based Methods for Fault Diagnosis: Some Guidelines. *Transactions of the Institute of Measurement and Control*, 17(2):73-83.

Patton, R. Frank, P. and Clark, R. 2000. *Issues in Fault Diagnosis for Dynamic Systems*. Springer-Verlag, New York.

Pau, L.P. 1986. Survey of Expert Systems for Fault Detection, Test Generation and Maintenance. *Expert Systems,* 3:100-111.

Persin, S. and Tovornik, B. 2003. Real-Time Implementation of Fault Diagnosis to a Heat Exchanger. In Proceedings of *5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Washington, D.C., USA, 1053-1058.

Pouliezos, A. and Stavrakakis, G. 1994. *Real Time Fault Monitoring of Industrial Processes*, Kluwer Academic Publishers.

Preston, S. Chapman, C., Pinfold, M., Smith, G. 2005. Knowledge Acquisition for Knowledge-based Engineering Systems. *International Journal of Information Technology and Management*, 4(1):1-11.

Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning*, 1.

Quinlan, J. R., Kodratoff, M. and Bythe. 1987. Generalization and Noise. *International Journal of Man-Machine Studies*, 24:101-123.

Rengaswamy, R. and Venkatasubramanian, V. 1993. An Integrated Framework for Process Monitoring, Diagnosis, and Control using Knowledge-Based Systems and Neural Networks. In IFAC symposium*, On-line fault detection and supervision in the chemical process industries,* Eds. Dhurjati, P. and Stephanopoulos, G. Pergamon Press.

Saludes, S., Corrales, A., Miguel. L. and Peran, J. 2003. A SOM and Expert System Based Scheme for Fault Detection and Isolation in a Hydroelectric Power Station. In Proceedings *SAFEPROCESS 03*, Washington, U.S.A., 999-1004.

Schere, W. and White, C. 1989. A Survey of Expert Systems for Equipment Maintenance and Diagnostics. In *Knowledge-based system Diagnosis, Supervision, and Control* ed. Tzafestas, S. Plenum Publishing Inc.

Shneiderman, B. 1992. *Designing the user Interface*: Addison-Wesley Publishing Company, Inc.

Soliman, A., Rizzoni, G. and Kim, Y. 1998. Diagnosis of Automotive Emission Control System Using Fuzzy Inference, In IFAC Symposium, *Fault Detection, Supervision and Safety for Technical Processes*, Kingston Upon Hull, UK, 26-28 August 1997, 715-720.

Su Hua and Chong Kil-To. 2006. Neural Network Based Expert System for Induction Motor Faults Detection. *Journal of Mechanical Science and Technology*, 20(7):929-940.

Tzafestas, S.G. 1989. System Fault Diagnosis Using the Knowledge-Based Methodology. Eds. Patton R.J., Frank, P.M. and Clark, R.N. *Fault Diagnosis in Dynamic Systems, Theory and Application*, Prentice Hall.

Tzafestas, S.G. 1994. Second Generation Diagnostic Expert Systems: Requirements, Architectures and Prospects. *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, SAFEPROCESS 94, 1-6.

Ursino, D. 2002. *Extraction and Exploitation of Intentional Knowledge from heterogeneous Information Systems: Semi Automatic Approaches and Tools*, Springer Verlag.

Wang, S. K., Luo, M.J. and Shiech, H.Y. 2006. An On-line Failure Identification Method for Rotor Systems, *Proceedings of 25th IASTED International Conference MIC,* Lanzarote, Spain, 25-30.

Wagner, W. P., Otto, J., Chung, Q.B. 2002. Knowledge Acquisition for Expert Systems in Accounting and Financial Problem Domains. *Knowledge-Based Systems*, 15(8):439-447.

Wagner, W.P., Chung, Q.B., Najdawi, M.K. 2003. The Impact of Problem Domains and Knowledge Acquisition Techniques: A Content Analysis of P/OM Expert System Case Studies. *Expert Systems with Applications*, 24(1).

Waterman, D. A. and Newell, A. 1971. Protocol Analysis as a Task for Artificial Intelligence. *Artificial Intelligence*, 2:285-318.

Widman, L., Loparo, K. and Nielsen, N. 1989. *Artificial Inteligence, Simulation and Modeling*, Wilsey Inc.

Wu, W.Z., Zhang, W.X., Li, H.Z. 2003. Knowledge Acquisition in Incomplete Fuzzy Information Systems via the Rough Set Approach. *Expert Systems*, 20(5):280-287**.**

Xing, H., Hung, S. and Shi, J. 2003. Rapid Development of Knowledge-Based Systems via Integrated Knowledge Acquisition. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 17(3):221-234.

Xiong, N., Litz, L. and Ressom, H. 2002. Learning Premises of Fuzzy Rules for Knowledge Acquisition in Classification Problems. *Knowledge and Information Systems,* 4(1):96-111.

Yusong, P., Hans, P., Veeke, M. and Lodcwijks, G. 2006. A Simulation Based Expert System for Process Diagnosis, In Proceedings of *EUROSIS 4th International Industrial Simulation Conference (ISC 2006)*, June 5-7, Palermo, Italy, 393-398.

Yu, Q., Xiuxi, L., Yanrong, J. and Yanqin, W. 2003. An Expert System for Real-Time Fault Diagnosis of Complex Chemical Processes, *Expert systems with applications,* 24(4):425-432.

Zadeh, L.A. 1994. Fuzzy Logic, Neural Networks and Soft Computing, *Communications of the ACM*, 37:77-84.

Zhang, X., Polycarpou, M. and Parisini, T. 2002. A Robust Detection and Isolation Scheme for Abrupt and Incipient Faults in Nonlinear Systems, *IEEE Transactions on Automatic Control,* 47(4):576-593.

TMRF e-Book
Advanced Knowledge Based Systems:
Model, Applications & Research
(Eds. Sajja & Akerkar), Vol. 1, pp 74 – 100, 2010

Chapter 5

Development of Knowledge Based Intelligent Tutoring System

Sunandan Chakraborty, Devshri Roy, Anupam Basu*

Abstract : Intelligent Tutoring Systems (ITS) are computer-based tutors which act as a supplement to human teachers. The major advantage of an ITS is it can provide personalized instructions to students according to their cognitive abilities. The classical model of ITS architecture has three main modules – domain, model, student model and teaching model. In this chapter, we have discussed the recent developments in those areas and tried to address the important issues involved with them. We have developed an ITS. We have used two major data structures (a tree and a directed graph) to represent the domain knowledge within the domain model. The domain model also consists of a repository which is a collection of study and test materials. The contents in the repository are annotated using various informative tags for intelligent retrieval of them for the students. A fuzzy state based student model is used to represent a student's cognitive ability and learning behavior. This is an important task as the adaptation techniques to be followed by the system are dependent to the student's ability. The actual adaptation techniques are planned and executed by the control engine, which is the teaching model of the system. The control engine communicates with the domain and the student model and takes decisions using fuzzy rules. We have evaluated our ITS and found quite encouraging results.

Keywords: Intelligent tutoring system, domain knowledge base, fuzzy based student model, teaching model

1. INTRODUCTION

One of the most important challenges faced by most of the developing and the underdeveloped countries is the spread of education among all. As an example, in India the literacy rate was about 65% and in the state of West Bengal it was 69.22% in 2001 (www.censusindia.net). A plausible reason behind this low literacy rate is attributed to the need of proper schools, proper infrastructure and poor teacher to student ratio (http://education.nic.in/stats/detail/18.pdf). The said perspective vindicates the

* *{sunandanchakraborty, droy.iit, anupambas}@gmail.com*
Indian Institute of Technology, Kharagpur, India

necessity of developing an alternate attractive, affordable and effective teaching platform that will capture the attention of children. In this scenario an Intelligent Tutoring System (ITS) can be quite relevant. An ITS can to a large extent address the issue of unavailability of skilled teachers. Intelligent Tutoring Systems (ITS) are computer-based tutors which act as a supplement to human teachers. The Association for the Advancement of Artificial Intelligence (AAAI) defined ITS as,

An intelligent tutoring system is educational software containing an artificial intelligence component. The software tracks students' work, tailoring feedback and hints along the way. By collecting information on a particular student's performance, the software can make inferences about strengths and weaknesses, and can suggest additional work (AAAI, AI Topics/Intelligent Tutoring Systems).

Ideally, an ITS tries to simulate a human teacher and sometimes it may prove to be more advantageous than its human counterpart. One of the main advantages of ITS is individualized instruction delivery, which means the system will adapt itself to different categories of students. A real classroom is usually heterogeneous where there are different kinds of students, from slow learners to fast learners. It is not possible to provide attention to them individually, thus the teaching may not be beneficial to all students. An ITS can eliminate this problem, because in this virtual learning environment the tutor and the student has a one-to-one relationship. The students can learn in her own pace. Another advantage is that using this system teaching can be accomplished with minimum intervention from the teachers. Therefore, ITS can be really effective in areas where there is dearth of trained teachers.

The functionalities of an ITS can be divided into three major tasks, (1) organization of the domain knowledge, (2) keeping track of the knowledge and performance of the learner, (3) planning the teaching strategies on the basis of the learner's knowledge state. Hence, to carry out these tasks, the ITSs have different modules and interfaces for communication (Wenger, 1987; Freedman, 2000a; Chou et al, 2003). The different modules are:

- Domain model
- Teaching model
- Student model
- Learning environment or user interfaces

Two major issues related to an ITS are "what to teach" and "how to teach" (Murray, 1999; Ong & Ramachandran, 2000). The domain model deals with the "what to teach" part , where as the teaching and the student model are concerned with the "how to teach" part. The main objective behind an ITS is its intelligence or adaptive teaching methods. Adaptation means that the system moulds itself to cater to different needs of different students. Among the two functions of ITS described earlier, adaptation is a part of the "how-to-teach" part. To decide on *how to teach,* the system needs to judge the student's knowledge state and her preferences. The tool which mediates between the system and the student is the *student model*. Thus, a robust, flexible and comprehensive *student model* is required to build a real adaptive Intelligent Tutoring System. The domain model is also important as it is the representation of the domain knowledge. Good design principles adapted in designing the domain model would help the system in selecting the proper methods for teaching. This would also help the system in the search for alternative teaching plans when a particular method fails to produce success. Finally, the most important part of an ITS is the *teaching model*. This module is the centre of the whole system. It communicates with the other modules and does the entire decision making.

A knowledge base ITS can only be effective if the teaching model plans the teaching strategies properly. A major drawback of any knowledge base ITS is writing rules for generating teaching strategies. It solely depends on the knowledge of the expert. Another problem is that often experts themselves disagree and can write different rules for the same given problem. One other drwaback of a knowledge base ITS is building an efficient and complete student model. The adaptation technique which changes dymnamically for different kind of students depends on the student model. Again the development of the domain knowledge base is a very tedious job. It can only be developed by a

domain experts. Development of domain knowledge base is very costly in terms of time and effort required.
Most of the existing knowledge base ITSs were restricted to a single domain. So, changing them or reorganizing them to teach different subjects were either impossible or required the intervention of the system developers. Again, attempting to design a domain independent system has got its own problems. A procedure or a strategy to teach Physics may not be successful when used to teach History (Freedman, 2000b). A single teaching method would not work in a multi domain teaching environment. Another disadvantage is that the delivery medium of all the existing systems was also restricted to usually one language. This may prove to be a serious drawback in a cosmopolitan society.
Reviewing the state-of-the-art we found some general drawbacks in the existing systems. The drawbacks for each module are summarized below:

Domain Model

- Domain knowledge representations are not generic (Baffes et al, 1996; Khuwaja et al, 1994). For example the ITS Andes (Conati et al, 2002; Gertner & VanLehn, 2000) is a domain dependent and its framework is dependent upon the domain. Different courses from different domains cannot be represented using the same framework. There is a need to develop domain independent representation scheme.
- In many ITS, Learning materials are implicit to the systems. Adding new materials into the system is cumbersome. Moreover, materials incorporated are homogeneous. Different formats like, animation or video files are not supported by many of the systems. Furthermore, documents supported in these systems are mostly monolingual.

Student Model

- Graphical probabilistic models such as Bayesian Networks are often used for student modeling (Conati et al. 2002,Vomlel 2004) which requires considerably expert modeling effort and hence very costly. There is need to give attention to build rapid, cost-effective means of developing these environment (Martin et. al., 2005, Kodaganallur et al. 2005, Aleven et al. 2006).
- Domain dependency was inherent in many of the existing student models. The design of the existing student models were either fully (Conati, 2002, Zhou & Evens, 1999a) or partially domain dependent (Zapata, 2004). Hence, configuring the corresponding ITSs to teach in other domains required major changes in the configuration of the student model.
- Configuration of student models, particularly in authoring system is often necessary. In various ITSs the configuration of student models involved complex operations. The teachers need to have advanced knowledge of programming to do those tasks (Baffes & Mooney, 1996). The modifications also involved the use of some terms, which can be difficult for teachers to interpret or perform (Zapata, 2004, Thompson, 1996).
- In most of the student models, different parameters to represent the student's cognitive ability were not used (Conati, 2002, Zapata, 2004, Baffes & Mooney, 1996). There can be various attributes describing a student's cognitive abilty, and their values can give a better approximation of the student's cognitive ability. Thus a monolithic representation of the student as used might not be effective.
- In some other cases the student model is not transparent to the students. A student's interaction with the system is very vital for her performance. The student at any point of time must know the state she is in. But mostly the student is unaware of her status. The outputs of

the student model should be presented to the student in a clear, easily understandable format and should contain the right level of details.

Teaching Model

- Usually the teaching models in ITSs cannot be modified. The teaching strategy defined in the teaching model is fixed and the changes can only be performed by the system developers. But the teaching strategies might require modifications. So there should be provisions for minor modifications in the teaching model.

From the above discussion, it is clear that there are scopes for further improvement in ITSs and its various modules. There is a need to develop ITS which should be able to reduce the drawbacks of the existing systems. We have developed an ITS whose design goals are as follows:

- The domain knowledge representation scheme should be generic so that a course from any domain can be configured in the system.
- The system should support learning materials of different types and formats, so that it can cater to diverse groups of students. The inclusion and modification of the materials should be simple. The materials should be reusable.
- The student model should be able to represent a student's cognitive ability and learning preferences. The model should consider the different parameters through which a student's capabilities can be estimated better. This should also help the system to analyze a student's drawbacks and adapt accordingly. The student model should also be independent and compatible with any domain.
- The student model should be transparent so that students can be aware of their performance.
- The teaching model should try to plan the best teaching method for each student. In case of a failure the module should be able to identify the situation and re-plan as necessary.
- There should be provisions for teachers to change the rules of teaching (teaching strategy) so that the system can be used in different situations.

The chapter is organised as follow. First, we review the related work done in the field of ITS in Section 2. In section 3, we discuss the overall architectures of our ITS. Next, in section 4, we present a detailed description of the various modules of the ITS. Section 5 presnts a case study. Section 6 concludes the paper.

2. LITERATURE REVIEW

In 1984, Benjamin Bloom defined the "two-sigma problem," which states that students who receive one-on-one instruction perform two standard deviations better than students who receive traditional classroom instruction (Bloom, 1984). It is impossible for any institution to provide personal teachers to each and every student. This drawback strongly supported the use of computers, as a replacement to human teachers. Motivated by the above cause, lots research groups started to work in this field and developed various systems with various features.

2.1 Earlier Systems

The first generation of computer assisted education tools were called, **Computer-Aided Instruction (CAI)** systems. One of the early examples of such a tutoring system is the system by Uhr in the year 1969 (Sleeman & Brown, 1982). This system generated problems on arithmetic and questions on

vocabulary. But main problem of this system were it had no user modeling or adaptation technique. However, some contemporary systems, like Suppes (1967), system by Woods and Hartley (1971) (Sleeman & Brown, 1982) could be called adaptive because here the problems were according to the user's performances. But the user model they used was quite primitive in nature. The model was only a parametric summary; the actual knowledge state of the user was not stored. These systems can be termed as the precursor to Intelligent Tutoring System (ITS).
In the meantime, another genre of tutoring system came up. These types of systems were called ***Drill and Test***. Here only problems were presented to the students in form of tests. The students on the other hand are provided with the test results. A simple variation of this system was the Adaptive Drill and Test. In this version instead of just presenting the problems, the student's performance and response were collected, tabulated and later used to select problems. Thus at this point of time, it was felt that the student needed to be considered as an important factor, and no longer predetermined rules will work. An adaptation technique was quite necessary to tackle all possible responses from the students.

2.2 Emergence of ITS

In 1982, Sleeman and Brown reviewed the state of the art in computer aided instruction and first coined the term **Intelligent Tutoring Systems (ITS)** to describe these evolving systems and distinguish them from the previous CAI systems. They defined ITS as being computer-based (1) problem-solving monitors, (2) coaches, (3) laboratory instructors, and (4) consultants. (Sleeman & Brown, 1982). And for the first time the use of Artificial Intelligence was seen, which made the systems "intelligent". At this point of time emerging trends in Artificial Intelligence were applied in these systems. With new AI techniques coming up it seemed that the computers were almost capable of "thinking" like humans. This motivated ITS research further. Application of AI in ITS made it possible to achieve the goals more easily. Other reasons which motivated ITS research were:

- Modular and fine grained the curriculum.
- Customized for different student populations.
- Individual presentation and assessment of the content.
- Collection of data which instructors could use to tutor and remediate students

In spite of these there were some typical drawbacks found in many early systems. These drawbacks included:

- Domain dependent. The targeted domains initially were (Sleeman & Brown, 1982)
  - Symbolic Integration [e.g. Calculus Tutor (Kimball, 1982)]
  - Electronic troubleshooting [e.g. SOPHIE (Brown et al, 1981)]
  - Mathematics [e.g. EXCHECK (McDonald, 1981)]
  - Medicine [e.g. GUIDON (Clancey, 1982)]
- System assumed much or too little of student's knowledge.
- Presented documents had the wrong level of details.
- Little interactivity, lead to limitations in student's expressivity.
- Inflexible.

2.3 Recent Systems

Since the emergence of ITS various systems were built using different techniques and with different objectives. In this section, we discuss about some of the recently built ITSs.

2.3.1 Andes

Andes (Conati et al, 2002; Gertner & VanLehn, 2000) is an ITS which was developed to teach physics for the students in Naval Academy. Bayesian networks were primarily used in Andes for decision-making. The major foci of the system are (1) Select the most suitable strategy for the student (2) Predict Student's actions (3) Perform a long term assessment of the student's domain knowledge.

Andes is a domain dependent ITS. Each problem in the system was broken into some steps and Bayesian network was formed using those steps as nodes. So, the problems were represented in the system as Bayesian networks. The Bayesian network would predict the most probable path for the student during a course. Each student could have different approaches to a problem, the network would be adjusted accordingly (the probabilities would change) and finally for a new problem it would predict the best strategy for the student. There is also a problem-solver in the system. This problem-solver partially or wholly solved a problem to help the students. The Bayesian networks had two parts: static and dynamic. The static part had Rule nodes and Context-rule nodes. The rule node represented general physics rules and had binary values, T and F. The probability P(Rule=T) was the probability the student could apply the rule properly in any situation. Initially these prior probabilities had values 0.5 but the authors claimed more realistic values could be obtained after a trial run in the Naval academy.

The dynamic part contained the Context-rule node as well as four other nodes: fact, goal, rule-application and strategy-nodes. The fact and the goal nodes were also binary. P(Fact=T) was the probability that the student knew the fact and P(Goal=T) was the probability that the student is pursuing the goal. There might be more than one way to reach the goal or the fact nodes and that lead to having so many parents. The conditional probabilities P (Fact = T| $parent_i$) represented the probability of reaching the fact from $parent_i$. The strategy nodes were there, where, students had more than one options to choose from. And the rule application nodes represented the children of those strategy nodes. The rule application nodes basically represented the different applications of the strategy nodes. The strategies were mutually exclusive; i.e. the student could choose one and exactly one strategy at a time. Thus if an evidence increased the probability of one of the strategies it would definitely decrease the others. The probability values (P (Strategy-node = {x: x ε {child1, …, childn}}) of these nodes would depend upon the number of children the node had. Finally the Rule-application nodes were the connectors between context-rule nodes, Strategy nodes, fact and goal nodes to new derived Fact and goal nodes. In other words, those nodes had a single Context-rule node and strategy node and one or more than one fact and goal nodes (preconditions) as parents, and children of those nodes included some facts and goal nodes (Conclusion). They had binary values and P(R-A = T) meant the probability of a student applying the parent Context rule to the preconditioned fact and goal nodes to get the derived fact and goal nodes. The probability values would vary with students and thus application of rules, choosing from alternate paths etc would depend upon each student. But how the probabilities were derived was not stated.

2.3.2 VisMod

ViSMod is another ITS which used Bayesian network (Zapata-Rivera et al, 2004). In the system the Bayesian network was divided into three levels. At the top most level the concepts (to be taught) were represented in a hierarchical manner. After that in the second level student's performance and behavior were described. Finally the third level nodes represented some analysis on the student's performance. Only the first level is domain dependent, whereas other two levels would remain same over different domains. Again student can observe only the top two levels of the Bayesian net. The third level is only visible to the teachers. During a course the probabilities in the second and third level of the Bayesian net changed according to the student's performance. Those probabilities made a change in the first level values. i.e. the probability values in the first level were directly dependent on the two lower

levels. After the probabilities were computed the most probable path along the first level was determined and the first node of the most probable path was chosen as the next step in the course.
The Bayesian networks were initially formed by human teachers using textbooks. The prior and conditional probabilities were set according to the level of the topic. That is each topic was classified by three categories, *beginner*, *intermediate* and *expert*. The probabilities will vary according to this categorization. The initial prior probability values were set by conducting a manual session with the students. The Bayesian networks were constructed using concepts from the topic in concern. Suppose a concept has two parents, which means that the parents have some influence on their children. In other words evidence of knowing the parent increased the probability of knowing the children. There was a threshold value, determined by domain experts, which depicted the highest probability a student could reach in knowing the child concept. There was another parameter called the *weight* which defined the degree of influence the parent had on the child. The weights were calculated by subtracting the base value between the parent and the child from the threshold value and dividing it by the number of parents the child had.
The conditional probabilities were calculated from the above information. The conditional probability P(child-concept = *knows* | $parent_1$ = *knows*,…, $parent_n$ = *knows*) = base value (1/n) + $\sum$ $weight_i$. Each node in the network could have two values, *knows* and *doesnot-knows*. If one of the parent concepts was not known by the student then that weight would be equal to 0. In case two parents had equal values of conditional probability then expert's choice would break the tie. The Bayesian network for a topic was first constructed by making the nodes as concepts and the edges as their dependencies. Then the conditional probabilities along the edges were calculated using the above procedure. That's how the whole network was built.

2.3.3 InterMediActor

InterMediActor (Kavcic et al, 2003) is an ITS which used fuzzy inference mechanism. It had a data structure called *navigation graph*. This graph determined which concept comes after which. When there were multiple choices, decisions were made using fuzzy rules. The fuzzy rules actually connected the student's capability and the nature of the concept in hand to decide whether the concept was suitable for the student or not. The student's information and other related information of the topic and concept were described in the system as fuzzy sets. The fuzzy rule base established a relation between them and helped to make decisions. The rules antecedent consisted of three parameters. Firstly the difficulty level of the topic, it had the values *easy, normal* or *difficult.* The values of these variables were dynamic. If in a topic students scored high marks consistently, the degree of difficulty would be decreased and in case the scores were low, difficulty would be increased (this was done using some fuzzy rules). The second parameter in the antecedent was the score in the final test. That variable could take values as *positive, negative* or *no mark.* The final parameter was, knowing the prerequisites. Its values could be *not, little, enough, well,* and *very well.* The fuzzy rules mapped those parameters to a consequent which was the recommendation level of the next topic, i.e. to determine whether the concept was suitable or not for the student. The consequent variable *level-of-recommendation* had values like, *learned, more recommended, recommended, less recommended or forbidden.* To defuzzify the consequent of the rules Centre of Maximum technique was used. The navigation graph, which represented all the topics and their dependency, was annotated by this crisp value of the level of recommendation. The annotated version of the graph represented the student's level of performance and understanding of the topic. The rules look like,

IF the student has *positive* mark AND the competence is *easy* THEN the concept is *more recommended.*

2.3.4 SQL-Tutor

SQL-Tutor (Wang & Mitrovic, 2002; Mitrovic, 2003) is an ITS, which as the name suggests is to teach SQL. Artificial Neural Network (ANN) was used in SQL-Tutor for decision-making. An agent was present who analyzed the student and selected an appropriate problem from the database. That agent was modeled using an ANN.

This ITS was developed to teach university students SQL. Here the solutions to the problems were represented in the system as constraints. Whenever a student submitted a solution the system calculated the correctness by comparing the number of constraints violated by the student. The next problem to be chosen or any other teaching decision to be taken depended on this information, how many mistakes or violated constraints the student has committed. To make this prediction the system used an artificial neural network (ANN). The ANN was a feed-forward network and had four inputs, one output and a hidden layer. Delta-bar-delta back propagation and linear tanh (LT) transfer function was used and the inputs consisted of different information relating to the student like, (1) Time needed to solve the problem, (2) The level of help provided to the student (3) The complexity of the problem (4) Also, the level of the student in hand.

In the output the ANN tries to predict the number of errors (i.e. the no. of constraints violated) will be committed by the student. This prediction is used to take the next teaching decision (like the next problem to choose from the problem database). However how the weights of the ANN were chosen and how exactly the ANN was trained were not explained clearly. About the performance of the system, the authors claimed that the ANN could predict the correct number of errors with an accuracy of 98.06%.

An added advantage of this system was it provided feedbacks to the student after checking her solution. The feedback might contain hints, partial solution or complete solution as required.

2.3.5 C++ Tutor

C++ Tutor is a rule-based ITS (Baffes & Mooney, 1996). Here the concepts of C++ were explained using some rules. These rules were in form of Horn sentences and were called the *Theory.* The problems were produced to the students in the form of feature vectors. The students were supposed to label the vectors choosing from a set of labels. An algorithm called *NEITHER* took these labeled vectors (student's solution) as input and modified the correct rule base, so that the modified rules implied the student's solution rather than the correct solution. This process is called *Theory-revision.* So now the modified rule base reflected the student's state of understanding, representing the student's correct knowledge as well as the misconceptions. After the theory-revision was complete the system tried to explain the bugs in student's concept by showing examples, which enumerated the areas where the student had gone wrong. This was done automatically by the system, by comparing the modified rules with the correct ones.

2.3.6 CIRCSIM Tutor

CIRCSIM is a dialogue based system which taught some topics on Physiology to medical students (Evens et al, 2001; Khuwaja et al, 1994). Here the problems were presented in the form of dialogues and the interactions with the students were also done in dialogues. In CIRCSIM the student model was divided into four components (1) Performance model (2) Student reply history (3) Student solution record (4) Tutoring history model.

In the performance model the student's performance was stored and analyzed. This assessment was done in four levels, *global,* which dealt with the overall performance of the student. *Procedure-level,* this is concerned with the particular problems the student has solved. *Stage assessment* checked the student's understanding about the different stages of Physiology in the problems. Finally, *local assessment* measured the student's perception in the Physiology variables that have been tutored. There were some rules which defined the priorities of the above components and the action taken. By priorities it is meant that even if the overall performance (component i) was poor for a student but the

reply history (component ii) was strong the system would treat the student as a competent student, i.e. component (i) had a lower priority than component (ii). Going by the rules the system would generate fitting dialogues to interact with the student. Thus the generated dialogues would depend upon the student model and would try to respond as correctly as possible. For example the system would try to give simpler problems (providing various hints if not solved correctly) to a student whose overall model was poor. And all these were done through dialogues. Other decision-making processes were also done using *simple rules*. It is stated that the rules could take consistent decision, considering all the information from the student model, but the exact methodology was not stated.

We have developed an ITS which has a flexible domain model. The domain model has a generic structure and courses from different domains can be built. It contains learning material in different languages. We found that several authors have explored the use of Bayesian belief networks to represent student models (Collins et al., 1996; Conati et al., 2002; Horvitz et al., 1998; VanLehn & Martin, 1997; Reye, 1996; Reye, 1998; Villano, 1992). Although existing configurations of student models can be represented using Bayesian belief networks in a solid and computationally tractable fashion, understanding the status of a Bayesian network can be difficult. It also requires considerably expert modeling effort and hence very costly. There is need to give attention to build rapid, cost-effective means of developing these models. There is a need to develop student model which should be easily understandable by students and teachers. Bull and Pain (1995) found that students seem to understand textually presented models. In order to build a flexible student model, we propose a fuzzy-state transition based mathematical student model, where a student's performance is represented by fuzzy-states and the transitions signify the student's progress. Here, some inputs are fuzzy variables and requires linguistic words to be given as input. The domain expert need not to provide the precise numerical value as a result authoring the student model becomes easier and requires less time and effort. The student model is transparent to the student. The overall architecture of the system is discussed in Section 3.

3. ARCHITECTURE OF THE ITS

We have built an ITS, which provides adaptive learning environment for the students by providing personalized instructions. It also exploits the use of multilingual and multimodal contents, which offers the students with a variety of materials, with different media and alternative modes of explanation. We adapted the classical structure of an ITS with four basic modules (Wenger, 1987; Chou *et al*, 2003) *domain model*, *student model*, *teaching model* and *student's interface*.

The domain model has a flexible structure and the different courses from different domains can be represented by using the same representation scheme. The *domain model* has two major parts. The first is called the *Domain Organization Model*, it is the knowledge base of the system. The other is the *Repository*, which stores the metadata annotated learning and test materials. *Student model* stores information about the student's performance state in terms of fuzzy variables, her preferences and her learning history. This module provides the basis of the adaptation policy employed by the ITS. The *control engine* or the *teaching model* is the pedagogical agent of the system and it executes the actual teaching process. In logical sense this module lies in the centre of the system and by communicating with the other modules it provides adaptive instructions to the students. It is rule-based, where the rules define the teaching strategy. The rules are authorable that enables the teaching model to be customized. The features of the authoring tool have been extended to support the modification of the teaching model. A domain expert can utilize this facility and apply her skills in reorganizing the teaching method to suit in different environments. *Student's graphical user interface* (GUI) is the gateway of the system for communicating with the student. All the learning materials and test sets are presented to the student through this interface and test results are fed back to the system for assessing the student. The overall architecture of the ITS is shown in Figure 1. We discuss its various modules and their functionalities.

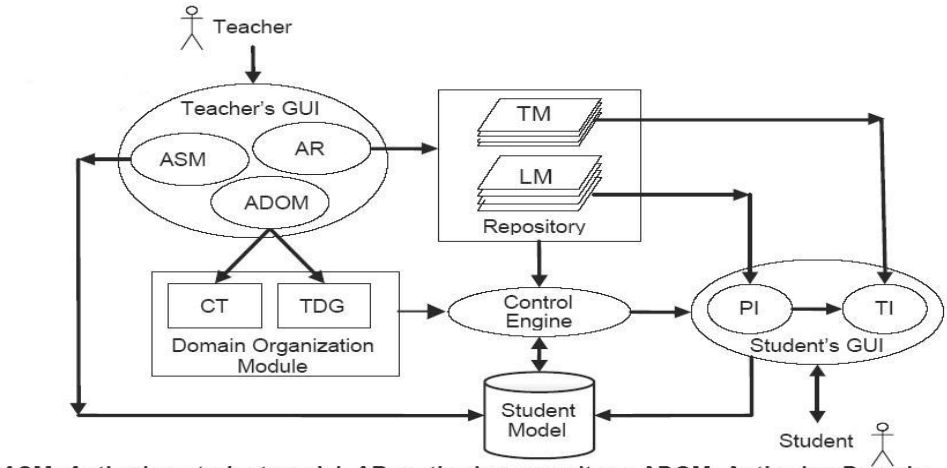


Figure 1. Overall architecture of the ITS

3.1 Domain Model

The domain model is the knowledge base of the system and it organizes the course structure, its various components and the relationship among the components. This model mainly deals with the what-to-teach part of an ITS (Wenger, 1987; Murray, 1999).
The knowledge base of our system has two main modules: *Domain organization module* and *Repository*. *Domain organization module* is concerned with the structure and organization of the course and its topics and relationship between the topics. It is the knowledge base of the system. The second module is the *Repository* which stores learning materials and also the test materials. Each learning material and test material is described by a set of metadata to provide mechanisms for finding and reusing of existing resource in the knowledge base.

3.1.1 Domain Organization Model

Domain organization module is a structural representation of the different courses. Two major data structures are used for this purpose are *Course Tree (CT)* and *Topic Dependency Graph (TDG)*.

Course Tree (CT): CT is a hierarchical structure of a course stored in the system. The root of the tree keeps the name of the course, and is called course node. Subsequently, the different sections/subsections and topics of the course are kept in lower parts of the tree. The leaf nodes of the tree are called topics and are the atomic teachable units. Topic nodes are also associated with different features like *Concepts (sub-topics), prerequisite topics, threshold score, difficulty,* and *importance*. A sample CT is shown in Figure 2.

Topic Dependency Graph (TDG): The nodes of TDG are constituted of the topics from the corresponding CT, whereas the edges in TDG depict 'prerequisite' relation between the nodes (topics). TDG drawn from the CT in
Figure is shown in Figure 3.

Every course stored in the system will have its own pair of CT and TDG.

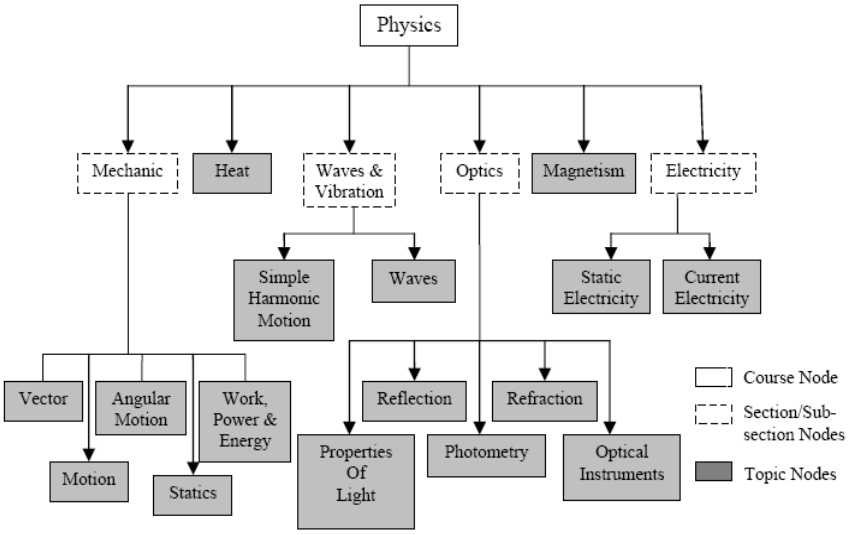


Figure 2. Course Tree for Physics

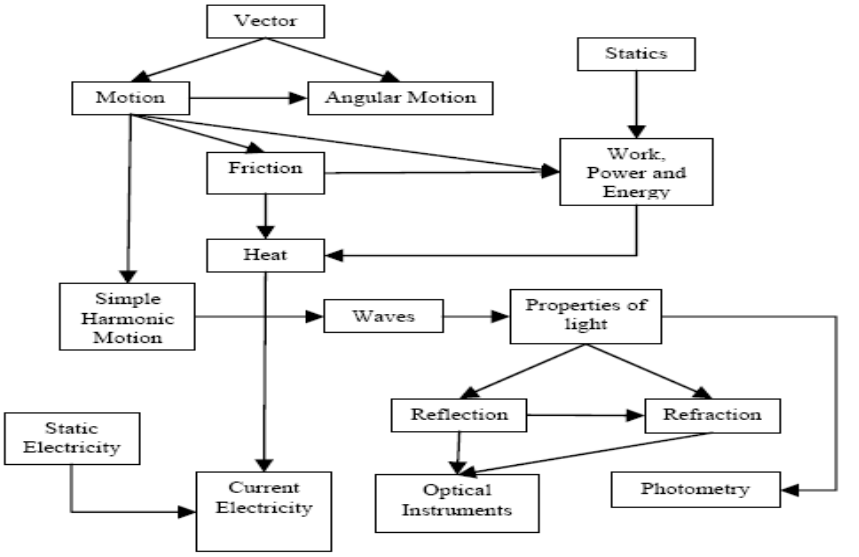


Figure 3. Sample TDG drawn from the CT in Figure 2

3.1.2 Repository

The Repository is a pool of learning and testing materials. For efficient access of materials from the repository the materials are tagged with various features. These features provide description of the documents, which helps the material selection agent of the system in efficient and customized retrieval of the materials for the students. In order to tag the study materials a subset of the standard set of tags specified by IEEE LOM is used. In addition some extra features are added locally to incorporate some system specific information in the document description.

3.2 Student Model

The most important task of an ITS is the application of its "intelligence". This means applying different methods of teaching to different students, without any intervention from the human teachers. In order to decide upon the method of teaching, the system must know about the nature and preferences of the student. Teaching can be most effective when the learner's cognitive ability and her behavioral pattern can be properly assessed from her learning outcome. Pointing out a student's deficiencies and concentrating on those areas is an important phase in any teaching process and can certainly result in a gain in her performance. In this teaching environment there is no other human involvement apart from the student. Hence, a tool or a utility is required within the system which will mediate between the system and the student. This tool should be able to assess the student properly and provide the decision-making agent of the system with all the relevant information, such as, the drawbacks of a student, which is required for adaptive teaching. This tool is called the *student model*. A robust, flexible and comprehensive student model is very vital for an adaptive Intelligent Tutoring System. Ideally a student model should keep the cognitive state of a student, her learning preferences, goals etc. In other words, the student model is an abstract representation of the student. Similar to normal teaching, where the human teacher reacts and changes the teaching method according to the feedback received from the student, the planning agent in an ITS must adapt itself and modify the teaching plan according to the feedback received from the student model. Our goal is to design such a student model which should be able to provide the system with all the required information. This information should be effectively used by the system's planning module to provide the students with personalized education, meeting the needs of the individual students. Thus, a well designed student model is necessary in meeting the objectives of an ITS.

The mostly used techniques for student modeling are Bayesian networks, Rule-based systems, Fuzzy logic, Dialogue-based systems etc. Andes (Conati et al, 2002) and VisMod (Zapata et al, 2004) are two ITSs which used Bayesian network to model a student. SQL-Tutor (Wang et al, 2002) modeled the student using artificial neural network. Fuzzy logic is another popular method used in student modeling (Ali et al, 2004; Kavcic, 2003; Weon, 2001; Xu, 2002). C++ Tutor (Baffes & Mooney, 1996) adapted a technique called theory refinement, which was also capable of capturing the students' misconceptions. Some other methods include feature-based modeling (Webb, 1997), machine learning methods, like, expectation maximization (Ferguson et al, 2006), reinforced learning (Beck & Woolf, 2000; Beck, 1997). Apart from the above techniques there are some works which focused on non-cognitive aspects of a student to model their learning styles. D'Mello (2005) used affect sensors to track the emotion of the learner and the system acted according to the students' state of mind.

We have adopted a fuzzy state based transition model for our student model. The transitions represent student's change of performance after a learning session. After a student covers a topic a state transition may occur or she might stay in the same state.

3.2.1 Student Profile

Student model should be a good approximation of a student and her learning progress. Therefore, our student model primarily consists of a student profile. In this profile a student is described using various parameters. Presently we are using only two aspects to measure the cognitive state of a student, namely *Comprehension-ability* (C) and *Problem-solving skills* (P). *Comprehension-ability* represents the student's ability to comprehend or understand a particular concept, whereas *problem-solving skill* indicates the student's capability to apply the concept which she has learnt particularly in problem solving. These two parameters collectively represent a student's overall aptitude. The values of *C* and *P* are taken to lie in the range [0, 1]. The student profile is a dynamic module, where, these values are updated every time a student appears in a test on a topic. So, at a particular instant the value of <*C*, *P*> pair signifies the current status of the student's cognitive state. The individual values of these parameters will help the system in assessing the student properly and act accordingly. For example, if a

student has a high value in comprehension ability but relatively low score in problem-solving skill, the system will try to provide her with learning materials, which will enhance her problem-solving skills. In the student profile two parameters are used to evaluate a student, but it can be easily extended to accommodate other parameters. The inclusion of some extra parameters will help to capture some other aspects of a student. Evaluation of a student will be stronger as various such parameters will give a better analysis of the student's cognitive state. Furthermore, adaptation will be more effective as the system will have a more accurate insight of the student's weaknesses and strong points.
In addition to *Comprehension-ability* and *Problem-solving skills*, the student profile also keeps records of some other information related to a student, for example, the types of documents preferred by the student. A particular student may have her own preferences, which might help her to learn better. This information kept in the student profile helps in the planning process, where the system will try to select a set of suitable materials to teach a topic better. Apart from this, the student profile also keeps other records, such as, sequence of topics covered and the student's performance in them, all the materials studied during the learning phase, detailed test results etc. The whole student profile is dynamic. After each performance evaluation of the student, the various fields of the student profile are updated from the evaluation results.

3.2.2 Fuzzy-state based State Transition Model

We have adopted a fuzzy-state based transition model for student modeling. In this model, each state represents the current status of the student's progress in a course including her performance. The transition from one state to another signifies the student's improvement or deterioration from her previous performance. Hence, the overall traversal in this model will depict the student's net performance throughout the course. In this system more than one course can be configured. For different course there will be separate transition models. Thus, the performance of a student in a particular course will not affect the decision-making in other courses.
The transitions help the planning agent of the system to take appropriate decisions during the planning process. A transition showing no improvement from the previous performance signifies the failure of the existing mode of teaching. Thus, the agent will try to select an alternative mode of teaching, such as selecting an alternative set of materials, to help the student to learn better. Similarly, improving transitions express the success of the current teaching method, which leads to the retaining of the same. We now illustrate the transition model in detail.
The student model is defined as a 3-tuple transition system, $<S, I, \delta>$. Where,

- S is a set of states,
- I is the input set, where $i \in I$ is a real number,
- δ is the transition function.

Definition of States: There are several parameters for pedagogical modeling of a student during a learning process, but in this work we consider two parameters,

- Performance of a student, measured through her ability to comprehend and her problem solving skills.

- Coverage: The topics covered by a student

Consequently, the state of a student is a composite state
$S = S1 \times S2$,
Where S1 denotes the set of performance-states and S2 denotes the set of coverage-states.
Therefore, set of states S is the set of composite states, $S = \{Si\}$,

where each composite state Si (after the i^{th} input) is a tuple <$Performance_i$, $Coverage_i$>

$Coverage_i$ is the list of topics that a student has covered successfully. Successful coverage of a topic implies that the student has scored above the set threshold value in the test conducted.
$Performance_i$ represents the student's performance. We propose a fuzzy state representation, where the performance-state of a student is represented in linguistic terms, represented by fuzzy sets. $Performance_i$ is denoted as a fuzzy term as defined below:
$Performance_i$ = {Excellent, Very Good, Good, Average, Poor}
The advantages of using fuzzy sets are, the use of linguistic terms are natural and easier to understand for the users (both students and teachers) and they can manage the uncertainties involved in these parameters.

Computation of Fuzzy State: For a student k, all the state information is stored in the student profile. At any "Present State" a student has already covered some of the topics successfully and has demonstrated a particular level of performance, which might have been modified through her last test. Computation of $Coverage_k$ for a student *k* is simple as it is augmentation to a list. The list is augmented when a student clears the test corresponding to a topic with a score greater than the set threshold score.
For computation of $Performance_k$, we are using the two parameters kept in the student profile, *Comprehension-ability* (C_k) and *Problem-solving skills* (P_k)**.** The values of C_k and P_k lie in the range [0, 1]. These values are computed every time a student appears in a test on a topic. After each update of the <Ck, Pk > pair for a student in her student profile, the fuzzy state is computed as follows (Weon & Kim, 2001):
Assume that, after the student *k* has covered the j^{th} topic her value of comprehension-ability and problem-solving skills have been C^k_j and P^k_j . respectively.

$$Now,\ let\ v = \frac{(C^k_j + P^k_j)}{2}$$

The membership functions of the fuzzy states are defined as:

$$\mu_{Excellent}(v) = 1 \quad if \quad 0.9 \le v \le 1$$
$$= 0 \quad otherwise \qquad (1)$$
$$\mu_{VeryGood}(v) = v$$
$$\mu_{Good}(v) = 2v \quad if \quad 0 \le v < 0.5$$
$$= (2-2v) \quad if \quad 0.5 \le v \le 1$$
$$\mu_{Average}(v) = 1-v$$
$$\mu_{Poor}(v) = 1 \quad if \quad 0 \le v \le 0.1$$
$$= 0 \quad otherwise$$

The state with the highest membership will be taken as the performance state. The results of the test taken by the students after each topic serve as the input to the state machine. The input value to the state machine is calculated from the test results using the following parameters:

- Correctness of the answer (c)
- Response Time (t)

The correctness value is a binary value for the questions of objective nature. For subjective questions the teachers need to evaluate them manually and feed in the performance score along with the recorded

response time. The score for these questions are scaled down, so that they lie between [0,1]. The response time is the time taken to answer each question. The teacher puts a threshold time (*t´*) for each question (in min.). If the student answers correctly within *t´* then she gets full credit. Then from *t´* to *3t´* the value diminishes according to Eq. 2. Finally, after *3t´* the credit becomes 0. The score for each question, is calculated using the following formula.

$$\begin{aligned} \alpha &= c \text{ if } t \le t' \\ &= c \times \left(1 - \left(\frac{t - t'}{3t' - t'}\right)^2\right) \text{ if } t' < t \le 3t' \\ &= 0 \text{ if } t > 3t' \qquad \textit{where } t' \textit{ is the threshold time set for the question} \end{aligned} \tag{2}$$

where *c* is the correctness of the answer, having value 1 for a correct answer, 0 otherwise, *t* is the response time, and *t´* is the threshold time.
Input value surpassing the threshold value of a topic indicates a successful completion of that topic. This leads to the augmentation of this topic in the *Coverage* part of the state. Again, the input value is combined with the previous performance state to compute the next performance state.

3.3 Teaching Model

In any ITS, a teaching agent is required which will control and conduct the teaching processes of the system. This agent is the nucleus of the ITS which communicates with the other modules and plans the teaching strategies to be taken for individual students. It, along with the student model defines the "how-to-teach" task of the ITS (Wenger, 1987; Murray, 1999). Among the various tasks it needs to carry out, one of the most important one is to plan the sequence of the various curriculum and course elements to teach a particular course (Brusilovsky & Vassileva, 2003). Most of the planning is done separately for individual students. Thus, the communication with the student model is essential for any decision-making done by this module.
In our system, the overall task of the teaching model is divided into a couple of smaller assignments or sub-tasks. Depending on the data about the students' state available in the student model the teaching model engine –

- Traverses the Topic Dependency Graph to select the most appropriate topic sequence separately for each student,
- Selects the most suitable set of contents for the chosen topic (by the topic planner), from the Repository for a particular student.
- Collects and analyzes the feedbacks (test results) from the students and plans how to improve upon the drawbacks found in the feedbacks

The first task is done by the module *Topic Planner*. The second task is done by the *Material Selection Module*. A third module within the teaching model, called *Result Analyzer* assesses the student's result in the recent test and performs some tasks as well as decides on certain issues. These include updating the student model according to the performance of the student, checking the performance and deciding whether she needs to undergo repetition of the whole topic or some parts of it. This module also checks whether the current teaching plan is successful or needs alterations. To summarize the objective of this module, it can be stated that it deals with the dynamic adaptation of the system. If any decision-making does not prove to be effective this module identifies those situations and produces

alternative plans. It also performs some post-test operations such as updating the databases. It takes the results from the testing interface and analyzes them. Then it updates the student model and signals the control engine (TP and MSM) for the necessary changes.

3.3.1 Topic Planner

Numerous topics constitute a course and studying each and every topic will eventually cover the whole course. These topics can be ordered or sequenced in various ways, where each sequence will represent a different way of covering a course. Each sequence represents a separate teaching plan. Different plans can be more effective in teaching as different students can be offered different teaching plans. In a classroom only a single plan can be followed for teaching and all the students (having different capabilities and objectives) have to follow the same plan. Thus, in a classroom the method of teaching may not be effective for all the students. The objective of the Topic Planner is to search for all the possible plans and work out the most suitable one for each student, using which it is assumed that the student will learn efficiently.

Linearization of Topic Dependency Graph: Often a topic is related to the other and coverage of a topic is not possible without the knowledge of its dependent topic. This relation or the dependency is called *prerequisite-of* relation. If a topic is a prerequisite of another topic, the former should always be taught before the latter. In case in a sequence of topics, a prerequisite topic occurs after its dependent topic, the sequence will be an *inconsistent* sequence. The topic planner searches in the TDG for the prerequisite relations between the various topics and plans consistent sequences. These sequences are followed by the students to study a course. Different students may be offered different sequences.

We define three states for a given topic. The states are *learnt, ready-to-learn* and *not learnt*. As the names suggest the state *learnt* means that the topic is taught to the student and he learnt the topic. The state *not learnt* means that the topics is not taught to the student. A topic in *ready-to-learn* state depicts that although this topic is not yet studied by the student but all its prerequisites have been covered by her and this topic now can be studied. The difference between the *'not learnt'* and *'ready-to-learn'* topics are that, although both kinds of topics are not studied by the student, a *'ready-to-learn'* topic is permitted for teaching in the current conditions. After each topic is successfully learnt by a student, the Topic Planner searches for all the *ready-to-learn* topics. Among the searched topics it tries to compute the most suitable one as the next topic to be taught (the selection process is discussed later in this section). This process is repeated after each topic is learnt until the last topic is covered. This process of searching an appropriate and consistent sequence of topics for a student is called the *Linearization of the TDG.*

Figure 4 shows a section of the TDG of the course Optics. Root is a dummy node whose children are the possible starting topic (which has no prerequisites) of the course. The leaves are the final topics of the course and these topics are not prerequisites of any other topics in the course. In this tree every path from the root to any leaf is valid sequence covering all the topics in a course and the course can be taught by traversing from the root to any of the leaf. All the prerequisites of a topic (node) lie in the path between the root and the node's parent. This makes all these paths to be consistent. The task of the Topic Planner is to construct this tree and check which of the path will help the student to learn most effectively. However, the whole tree is not constructed by the Topic Planner in one go. The tree is expanded dynamically and only in one direction, and the other parts are pruned. This saves computation time as well as memory space. We will now discuss about how the tree is expanded and traversed, in order to cover or learn the course.

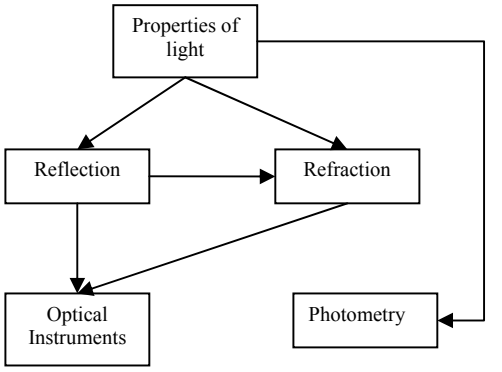


Figure 4. A section of the TDG of course Optics

According to the TDG (Optics) in Figure 4, 'Properties of Light' is the only topic without any prerequisites. Therefore, the starting topic will be 'Properties of Light'. After the successful completion of this topic there are two choices. The student can either study 'Reflection' or 'Photometry' next. These are the only two topics whose prerequisite is 'Properties of Light'. Thus the coverage of 'Properties of Light' changes the status of those topics to *ready-to-learn*. The expanded structure of the tree is shown in Figure 5. Now the task of this module is to decide which topic between the two will be better for the student to study next.

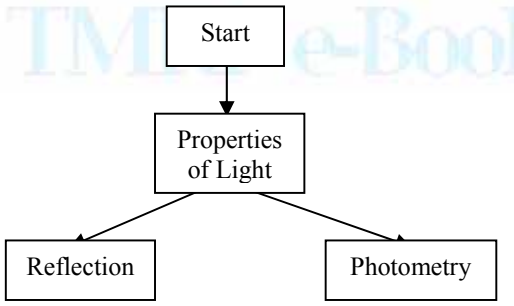


Figure 5. Initail expantion of the tree

In order to take such a decision the Topic Planner compares all possible topics (in this case 'Reflection' and 'Photometry) with the student's current learning state, obtained from the student model. Consulting the student model, each topic is assigned a *suitability value.*. This value expresses the suitability of a topic for a particular student. Now, the topic with a higher value of this *suitability value* will be chosen as the next topic. Say, in this case 'Reflection' had a higher value and was selected.

The suitability value of the *ready-to-learn* topics will be calculated again. Note that the suitability value of 'Photometry' might be different from the value calculated earlier, as student's current learning state is used to calculate this value, and the learning state might change after the coverage of 'Reflection'. Selection and expansion will continue until all the topics in the course are covered successfully. In Figure 6 a possible expansion of the tree is shown.

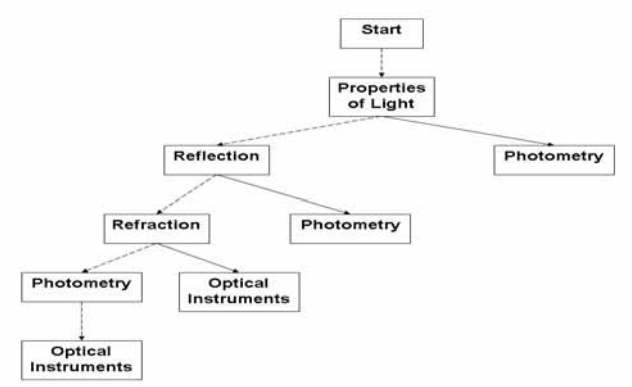


Figure 6. A possible exapansion of the tree

Methodology for Linearization: The decision making in this module, like the evaluation of the suitability values of the topics are based on fuzzy rules. There is a rule base consisting of various fuzzy rules. Tasks, like how to select a topic for a student or how to plan the course of teaching for a particular student are defined by these rules. These rules define the teaching strategy to be taken. Suitability value of a topic is the degree of appropriateness between this topic and a student. Therefore, calculation of this parameter involves a comparison between the topic and the student's current state of knowledge and her cognitive ability. Topics are described using various attributes and are kept in the domain model and the description of the student and her cognitive abilities can be extracted from the student model. Some of these attributes are used to establish a relation between the topic and the student and calculate how suitable the topic is for the student. The fuzzy rules define such relations (Kavcic et al, 2003). The antecedent of the rules is constituted by these attributes, whereas the consequent is the *suitability value*. The various features used in the antecedent of the rules, along with their meaning and values are shown in .

Table 1. Each of these terms is a separate fuzzy variable, whose values are defined using specific fuzzy sets (Ross, 1997).

Table1. Antecedents of the Fuzzy Rules

Topic Attributes		
Name	Values	Meaning
Hardness	{Tough, Moderate, Simple}	Hardness of the material
Importance	{VeryHigh, High, Moderate, Low, VeryLow}	How important this topic is in this course
Scope	{Excellent, VeryGood, Good, Average, Poor}	How many opportunities or avenues does this topic open up in the course
Student Attributes		
Performance	{Excellent, VeryGood, Good, Average, Poor}	Overall performance of the student in this course up to this point
Prerequisite Performance	{Excellent, VeryGood, Good, Average, Poor}	Performance in the prerequisite topics of this topic
Interest	{VeryHigh, High, Moderate, Low, VeryLow}	How much interest the student has in this topic

The rules derive the *suitability value*. Fuzzy term 'Suitability' has fuzzy values "Very_High", "High", "Medium", "Low" and "Very_Low". The 'Suitability' value is defuzzified using *centroid* defuzzification method (Ross,1997). A typical rule is shown here.

> **IF** Performance is *Excellent* **AND** Pre-equisite_Performance is *Excellent* **AND** Interest is *VeryHigh* **AND** Hardness is *Simple* **AND** Importance is *VeryHigh* **AND** Scope is *Excellent*

The rule-base is editable. Thus, they can be constructed or modified by experienced teachers using the Rule Editor of the authoring system. The rule base contains various such rules. During a decision-making process the inputs are fired with each and every rule in the rule base to get the individual outputs. Then the outputs are combined to compute the final *suitability value*. Individual *suitability values* are obtained for all the topics in *ready-to-learn* state. Finally, the topic with the highest Suitability crisp value from the set of *ready-to-learn* topics is chosen as the next topic.
This process is continued till the *ready-to-learn* set is empty. This infers that all the topics in the course have been covered by the student.

3.3.2 Material Selection Module

In the previous section we discussed about how topics were selected and sequenced for individual students. However, actual teaching or studying involves selection and presentation of the study materials. Selection of materials is also a part of the personalization scheme of this ITS. There are various kinds of documents to teach the same topic. Some are relatively difficult than the rest and different documents some have different approaches of explaining the same concepts. The system must interpret the student's preference and ability to select the proper type of documents for her. This module searches the repository to construct a set of suitable documents to teach each topic for each student. After the *Topic Planner* selects a topic the *Material Selection Module* (MSM) is invoked. This module produces a set of documents which the student follows to learn the current chosen topic. The repository contains documents assocaited with a set of metadata. The metadata associated with the documents helps in retreiving the learning materials for the chosen topic.

3.3.3 Result Analyzer

The third task of the Control Engine is to analyze feedbacks obtained from the student's test results and perform some operations accordingly. This task is undertaken by the Result Analyzer (RA). After a session of learning through study materials the students must undergo a test. These tests are a reflection of the student's understanding in the topic. Using this feedback, the ITS gets an updated knowledge of the student's performance and uses this knowledge to take various decisions.
Preliminary task of the RA is to provide the results to the student model and update the various fields of the student profile. It supplies the test results to the student model so that it can compute the next state of the student's performance. Apart from these, RA scrutinizes the test results to obtain some analysis on the student's performance. This analysis finds out the various faults in the student' learning and some decisions are taken, so that the student can overcome her flaws. RA also checks whether a change in the teaching plan is necessary for a better outcome from the student. Thus, RA helps the system to adapt dynamically with the changing condition of the student. An earlier plan may not work for a student and she may get stuck at some point and cannot advance from there. Such a condition can be only detected from the feedbacks (test results) from the student. Thus, RA identifies such situations from the test results and works out methodologies to get out from those situations i.e. revise the teaching plan computed earlier.

4. CASE STUDY

The system was used in a real teaching and learning scenario. We organized a workshop for school students to learn basic 'Programming in C' using the ITS, *Shikshak*. Thirty three students from the different schools in the campus of IIT, Kharagpur participated in this workshop. The participants varied from 12 to 16 years in age. No students had any prior exposure to programming. The students studied various topics of C programming. First step was to configure the course in the ITS. Course Tree Editor was used to configure the Course Tree (CT) for this course. The configured CT is shown in Figure 7. As this course was meant for school students, only four basic topics were chosen. The topics were,

- Variables, data type and operators
- Loops
- Branching
- Array

The attributes and their values of these topics are shown in Table 2.

Table 2. Topics and the attributes

Topics	Hardness	Importance	Prerequisite topics	Concepts
Variables,Data types, Operators (V)	0.5	High	None	Variables, Arithmetic and logical operators, data types, printf, scanf
Loops (L)	0.8	High	V	For loops, while loops
Branching (B)	0.6	Moderate	V	If-else, switch-case, continue, break
Array (A)	1.0	Low	L	Linear array, string

Figure 7. Screenshot of the Course Tree Editor showing Course Tree constructed for the experiment

All the topics were furnished with the required details, such as *concepts, prerequisites, difficulty* etc. Accordingly the system drew the TDG from the prerequisite information. The drawn TDG is shown in Figure 8.

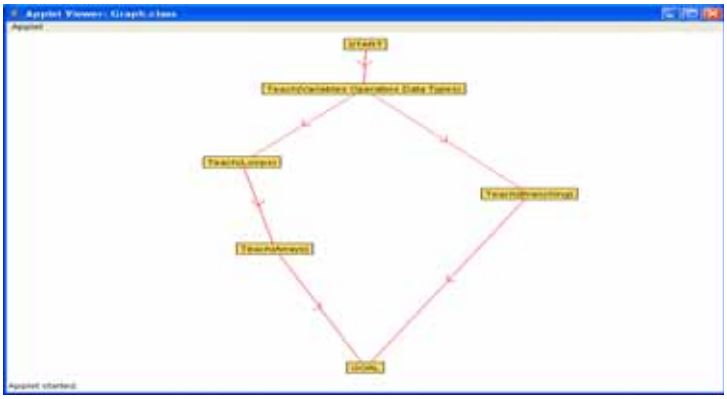


Figure 8. Screenshot of TDG drawn by the system from the CT in Figure7

The next step involved preparation of the repository. For each topic various materials were created or collected and incorporated into the repository. Care was taken to make sure the materials were heterogeneous in nature. Moreover, files with different presentation styles like *text-intensive*, *animation based* or *presentation type* of documents were also incorporated. Some materials, particularly Macromedia flash based animation based documents were built specially for this purpose. Screenshot for such a document is shown in Figure 8. Variety of documents enriched the repository and helped the *Material Selection Module* to select the appropriate type of materials from the different variety available. This served the basic purpose of the ITS, where unlike a classroom, students had the opportunity of studying from materials which suited them best. This was followed with the annotation of the materials. Annotation was done very carefully, as incorrectly annotated materials would lead to inconsistent retrieval from the repository and affect the system's performance. In order to test the students' knowledge various questions were also frames. The questions varied in terms of difficulty and focus (comprehension-ability or problem-solving skills)

Apart from domain model organization, some other preparations were required in the student model and the control engine. Student model was configured to accommodate the participating students. This initialization of the student model involves furnishing of some initial information about the students. These data were obtained from their school records and fed into the student model. In the control engine some rules were framed, using the Rule Editor. These rules defined the strategies to be taken by the Topic Planner to sequence the course for each student.

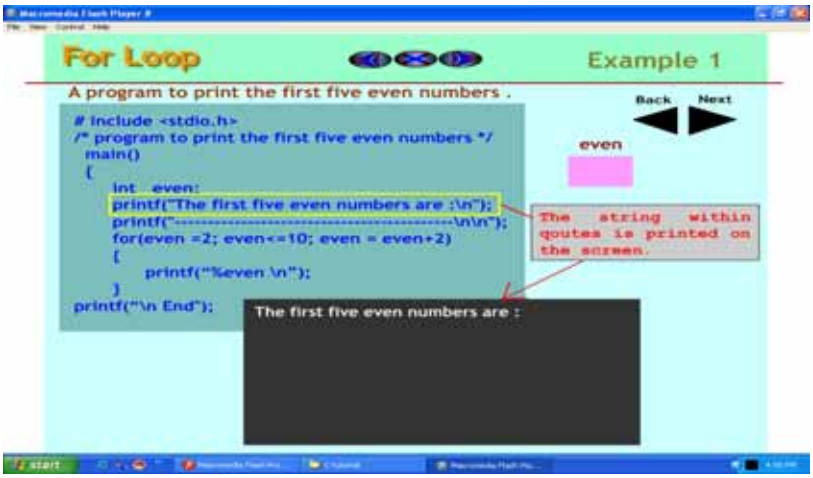


Figure 8. A sample learning material

The Topic Planner computes separate topic sequences for different students. In Figure we show the different sequences can be drawn for this course.

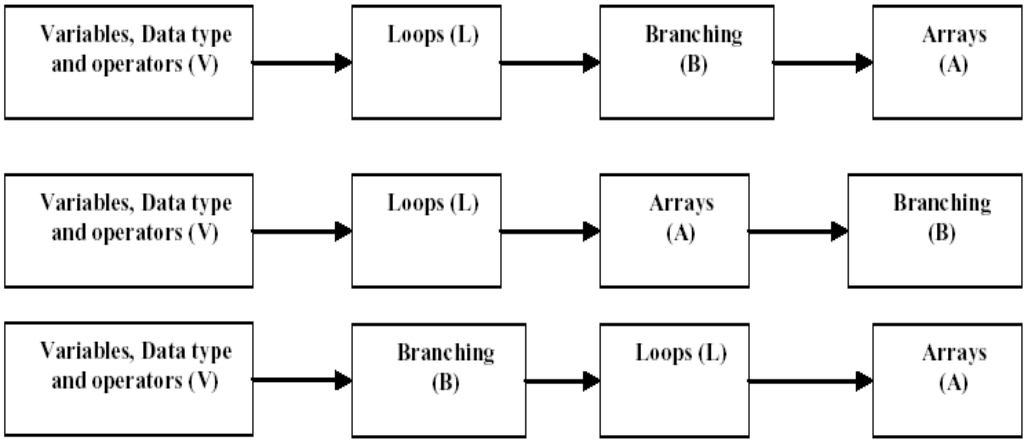


Figure 9. All possible sequences drawn from the TDG (Figure) used in the workshop

We present the percentage of students following each sequence in Figure 10. The "others" column collectively represents all the cases were sequences had to be re-planned for some students.

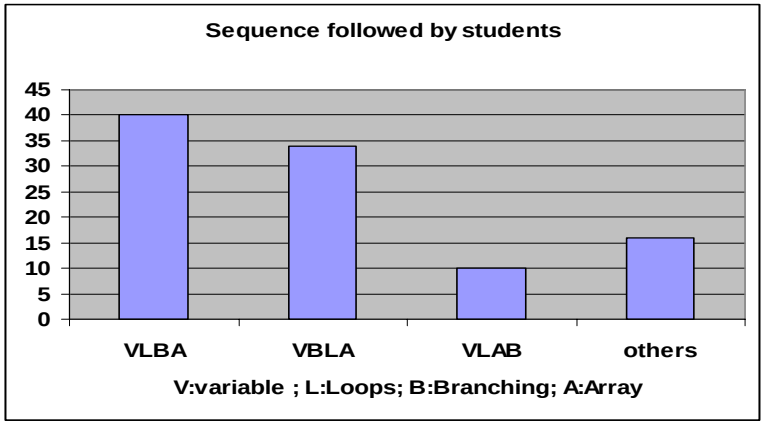


Figure 10. Percentage of each Sequence followed by students

We claimed that an important aspect of an ITS is that the students can learn and progress along a course at their own pace. The duration of the course was 15 days but many fast learning students completed the course before that. However, some of the students could not finish the course in 15 days. Perhaps 15 days were not sufficient and they required some more time. The detailed observations are shown in Figure11. The red (last three) columns signify that those students were unable to complete the course in 15 days.

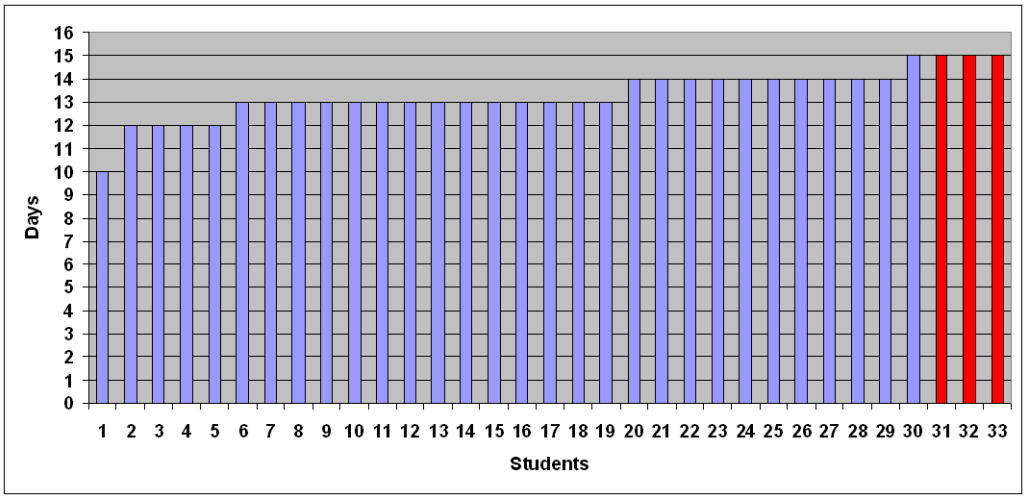


Figure 11. Days taken by each student to complete the course (the last three columns represent the students who attended the course for 15 days but could not finish)

5. CONCLUSION

In this chapter, we have discussed about the development of an Intelligent Tutoring System (ITS). We started the chapter by discussing about ITS and some its general features. In the following section we presented some previous works related to ITSs. Then, we described the complete system with a full

description of it's each module. Finally, we presented some results obtained during a field trial of the system.

The main objective of our work was to develop a system that can teach and deliver contents in a customized and adaptive manner. The system considers the cognitive ability of each student and devises teaching plans which would suit them best individually and maximize their performance. In summary, we can claim that the system exhibits moderately good performance. The main contribution of our works are summarized below:

- A generic and domain independent structure to store and represent the courses. Using this representation scheme various courses can be configured in the system. Simple courses for primary education to more complex courses of higher education can be configured using this scheme. The configuration of courses using this scheme is also simple and requires very little experience of computer usage from the teachers.

- A set of tags through which learning materials can be annotated. Most of the tags have been used from the IEEE LOM set of standard metadata for annotating learning materials. However, we have added some tags locally to make the annotation scheme compatible with our system.

- A fuzzy state transition based student model which is an abstract representation of the students using the system. This model also uses two important parameters to evaluate the cognitive ability of a student. They are comprehension ability which tests a student′s ability to understand a concept or a topic. The other is problem-solving skills, which checks a student′s ability to apply a concept in solving problems. The student model provides the basis of adaptation offered by the system.

- Another important contribution of this work is a fuzzy rule based teaching agent which controls the adaptive teaching process of this system. This agent communicates with the other modules of the system and produces customized teaching plans for the individual students. This module also keeps track of a studen's changing requirements along a course and adapts itself accordingly.

However, there are some limitations in the present system. One of the limitation is that there are no facilities to provide hints to students. This might help a student immensely in problem solving and avoid getting stuck into a problem. The present student model is capable of capturing two aspects of a student. These two parameters are not enough to capture a complete picture of a student's cognitive ability. The current system has been evaluated on the basis of the users who came from good urban schools. However, this system is to deploy in rural areas where the system can be quite relevant and might play an important role in improving the existing infrastructure in education. It is quite important to check how the system performs when users from those areas actually use the system. Therefore, a major future work is to deploy the system in various rural educational centers and evaluate its performance.

References

Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. 2006. The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley, & T. W. Chan (Eds.), in 8th International Conference on Intelligent Tutoring Systems (ITS 2006), Berlin: Springer Verlag, pp. 61-70.

Ali, M.S., Ghatol, A.A. 2004, A Neuro Fuzzy Inference System For Student Modeling In Web-Based Intelligent Tutoring Systems, International Conference on Cognitive Systems, New Delhi, December 2004.

Baffes, P., and Mooney, R. 1996. Refinement-Based Student Modeling and Automated Bug Library Construction. In Journal of Artificial Intelligence in Education, 7, 1 (1996), pp. 75-116,

Beck, J.E. 1997. Modeling the student with reinforcement learning. Machine Learning for User Modeling Workshop at the Sixth International Conference on User Modeling, 1997.

Beck, J.E., Woolf, B.P. 2000, High-Level Student Modeling with Machine Learning, in 5th International Conference on Intelligent Tutoring Systems, G. Gauthier, C. Frasson, K. VanLehn (Eds.): ITS 2000, LNCS 1839, pp. 584–593, Montreal, Canada, 2000.

Bloom, B.S. 1984. The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher* 13 (6):4–16.

Brown, J., Burton, R., deKleer, J. 1981, Pedagogical Natural Language and Knowledge Engineering Techniques in SOPHIE I, 11, and IIIH, Tutoring Systems, Sleeman et a1 (eds), Academic Press.

Brusilovsky, P., Vassileva, J. 2003. Course sequencing techniques for large-scale web-based education. In International Journal of Content Engineering and Lifelong Learning, Vol. 13, Nos. 1/2, 2003.

Bull, S., Pain, H. 1995. "Did I say what I think I said, and do you agree with me?": Inspecting and questioning the student model, *Proceedings of AI-ED'95 - 7th World Conference on Artificial Intelligence in Education*, AACE, pp. 501-508.

Chou, C., Chan, T., & Lin, C. 2003. Redefining the learning companion: the past, present, and future of educational agents, Computers & Education, v.40 n.3, pp.255-269, April 2003.

Clancey, W. J. 1982, GUIDON, In Barr and Feigenbaum (editors), The Handbook of ArtificiaI Intelligence, chapter Applications-oriented Al research: Education. William Kaufmann, Inc., Los Altos, 1982.

Collins, J. A., Greer, J. E., Huang, S. X. 1996. Adaptive Assessment using Granularity Hierarchies and Bayesian Nets. In Frason, C., C., Gauthier, G. and Lesgold, A., (eds.), Proceedings of Intelligent Tutoring Systems ITS'96. Berlin: Springer, pp. 569-577.

Conati, C., Gertner, A., VanLehn, K., & Druzdzel, M. 2002. On-line student modeling for coached problem solving using Bayesian networks. In *Proc. Sixth International Conference on User Modeling,* Vienna , 1997, pp. 231-242.

D'Mello, S. K., Craig, S. D., Gholson, B., Franklin, S., Picard, R. W., & Graesser, A. C. 2005. Integrating affect sensors in an intelligent tutoring system. In Affective Interactions: The Computer in the Affective Loop Workshop at International Conference on Intelligent User Interfaces 2005, San Diego, 2005, pp. 7-13.

Evens, M. W., Brandle, S., Chang, R., Freedman, R., Glass, M., Lee, Y. H., Shim, L. S., Woo, C. W., Zhang, Y., Zhou, Y., Michael, J. A., & Rovick, A. A. 2001. CIRCSIM-Tutor: An intelligent tutoring

system using natural language dialogue, in Proc. *Twelfth Midwest AI and Cognitive Science Conference,* Oxford, 2001, pp. 16-23.

Ferguson, K., Arroyo, I., Mahadevan, S., Woolf, B., & Barto, A. 2006. Improving intelligent tutoring systems: Using expectation maximization to learn student skill levels. In Proc. 8th International Conference on Intelligent Tutoring Systems, Taiwan, June, 2006, *Lecture Notes in Computer Science*, No 4053, pp. 453-462.

Freedman, R. 2000a, What is an Intelligent Tutoring System?, Published in Intelligence 11(3): pp. 15–16.

Freedman, R. 2000b. Plan-based dialogue management in a physics tutor. Proceedings of the Sixth Applied Natural Language Processing Conference. Seattle, WA, pp. 52-59.

Gertner, A., & VanLehn, K. 2000. Andes: A coached problem solving environment for physics. In *Proc. 5th International Conference. Intelligent Tutoring Systems,* Berlin, 2000, pp. 133-142.

Horvitz, E., Breese, J. S., Heckerman, D., Hovel, D., Rommelse, K. 1998. The Lumiere Project: Bayesian user modeling for inferring the goals and needs of soft-ware users. Fourteenth Conference on Uncertainty in Artificial Intelligence. San Francisco: Morgan Kauf-mann, pp. 256-265.

Kavcic A., Pedraza-Jimenez R., Molina-Bulla H., Valverde-Albacete F.J., Cid-Sueiro J., Navia-Vazquez A. 2003. Student modeling based on fuzzy inference mechanisms, *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, vol.2, no.pp. 379- 383 vol.2, 22-24 Sept. 2003

Khuwaja, Ramzan, A., Evens, M. W., Michael, J. A., & Rovick, A. A. 1994. Architecture of CIRCSIM-Tutor (v.3): A smart cardiovascular physiology tutor. In *Proceedings of the 7th Annual IEEE Computer-Based Medical Systems Symposium, Winston-Salem, NC, 1994*, pp. 158-163. IEEE Computer Society Press

Kimball, R. 1982, A self-improving tutor for symbolic integration. In D. Sleeman & J.S. Brown (Eds.), Intelligent Tutoring Systems, New York: Academic Press, pp. 283-308.

Kodaganallur, V., Weitz, R., & Rosenthal, D. 2005. A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms. International Journal of Artificial Intelligence in Education, Vol. 15, pp. 117-144.

Martin, B., Koedinger, K., Mitrovic, T., & Mathan, S. 2005. On Using Learning Curves to Evaluate ITS. Twelfth International Conference on Artificial Intelligence in Education, pp. 419-426, Amsterdam.

McDonald, J. 1981, The Excheck CAI System, in University-Level Computer-Assisted instruction at Stanford: 1968-1980, P. Suppes, ed., Inst. for Math. Studies in the Social Sciences, Stanford Univ., Palo Alto, Calif., 1981.

Mitrovic, A. 2003. An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education, 13*(2-4), 2003, pp. 171-195.

Murray, T. 1999. Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art. International Journal of Artificial Intelligence in Education (1999), 10, pp. 98-129.

Ong, J., & Ramachandran, S. 2000. Intelligent Tutoring Systems: The What and the How. http://www.learningcircuits.org/2000/feb2000/ong.htm

Reye, J. 1996. A Belief Net Backbone for Student Modeling, Proceedings of third International Conference on Intelligent Tutoring Systems, June 12-14, 1996, pp. 596-604.

Reye, J. 1998. Two phase updating of Student Models based on Dynamic Belief Networks, Proceedings of fourth International Conference on Intelligent Tutoring Systems, Aug 16-19, 1998, pp. 274-283.

Ross, T.J. 1997. Fuzzy Logic with Engineering Applications. McGraw-Hill, 1997.

Sleeman, D., & Brown, J. S. 1982. Introduction: Intelligent Tutoring Systems. *Intelligent Tutoring Systems*, D. Sleeman, J. S. Brown, Ed. Academic Press, 1982, pp. 1-11.

Thompson, J.E. 1996. Student Modeling in an Intelligent Tutoring System. *MS Thesis.*

VanLehn, K., Martin, J. 1997. Evaluation on an assessment system based on Bayesian student modeling. International Journal of Artificial Intelligence in Education, Vol. 8. pp. 179-221.

Villano M. 1992. Probabilistic Students Models, Bayesian Belief Networks and Knowledge Space Theory, Proceedings of the second International Conference on Intelligent Tutoring Systems, June 10-12, 1992, pp. 491-498.

Vomlel, J. 2004. Bayesian networks in educational testing. Int. J. Uncertain. Fuzz. Knowl. Based Syst. 12(Suppl. 1), 83–100 (2004)

Wang, T., & Mitrovic, A. 2002. Using neural networks to predict student's performance. In Proc. of International Conference on Computers in Education, 2002, pp. 969-973.

Webb, G. I., B. C. Chiu, and M. Kuzmycz 1997. Comparative Evaluation of Alternative Induction Engines for Feature Based Modelling. International Journal of Artificial Intelligence in Education 8. Amsterdam: IOS Press, pages 97-115.

Wenger, E. 1987. Artificial Intelligence and Tutoring Systems. Los Altos, CA: Morgan Kaufmann.

Weon, S., & Kim J. 2001. Learning achievement evaluation strategy using fuzzy membership function. In Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference, Reno, NV, 2001, October pp. 10–13.

Xu, D., Wang, H., Su, K. 2002. Intelligent Student Profiling with Fuzzy Models, In Proceedings of the 35th Hawaii International Conference on System Sciences, Hawaii, USA (2002), p. 81b.

Zapata-Rivera, D., Greer, J. 2004. Interacting with Inspectable Bayesian Student Models. International Journal of Artificial Intelligence in Education, Vol 14. pg., 127 – 168

Zhou, Y., Evens, M. W. 1999a. A Practical Student Model in an Intelligent Tutoring System, ICTAI.

TMRF e-Book
Advanced Knowledge Based Systems:
Model, Applications & Research
(Eds. Sajja & Akerkar), Vol. 1, pp 101 – 125, 2010

Chapter 6

Spatial Navigation in Virtual World

***Kanubhai K. Patel*[*], *Sanjay Kumar Vij*[†]**

Absract One of the most challenging and complex tasks when working with virtual worlds is the navigation process. This chapter presents the possibilities of using neural networks as a tool for studying **spatial navigation** within **virtual worlds**. How it learns to predict the next step for a given trajectory, acquiring basic spatial knowledge in terms of landmarks and configuration of spatial layout. In addition, how it builds a spatial representation of the virtual world, rather like a cognitive map. The benefits of this approach and the possibility of extending the methodology to the study of navigation in Human Computer Interaction (HCI) and other applications are described in brief. The study of computation models of navigation and the potential of using cognitive maps in the modeling of navigational processes are described. **Non-visual spatial learning** model is presented for the spatial learning through virtual world exploration. Different types of **locomotion** in virtual world with their constraints and benefits are discussed. Intended readers of the chapter would be the mobility trainer, architect, city planner, cartographer, psychologist, and game developer.

Keywords: Artificial Neural Network, Navigation Model, Spatial Cognition, Virtual World

INTRODUCTION

One of the elementary human needs is the need to know the world around us and to be able to freely navigate within this environment. Our daily life is occupied with (indoor as well as outdoor) activities moving from one location to another. We spend much of our time moving from one place to another within or between different environments (habitats). Survival of the individual is contingent upon adaptive skills to find, learn and return to specific places and often at specific times - such as the home or the working place - quickly and safely. These skills are summarized by the word navigation. Our navigation ranges from our daily way to work and may extend even to global travel.

When we move around new environments, we subconsciously build a mental image of the space we are in. This mental image is stored in the hippocampus, and is called a "**cognitive map**". The term "cognitive map" was first used by Tolman (1948) to describe a mental representation of spatial information used for navigation. Cognitive maps are cartographic illustrations of a person's internal

[*] Asst. Professor, School of ICT, Ahmedabad University, Ahmedabad – 380 009 kkpatel7@gmail.com
[†] Director (CE-IT-MCA), SVIT, Vasad-388 306 vijsanjay@gmail.com

representation of the spatial environment in which they live. However, sixty years later, we still don't have any hard answers about the structure of spatial knowledge.
Swarms of social insects also construct trails and networks of regular traffic via a process of pheromone laying and following. These patterns constitute is also known as a cognitive map in brain science. The main difference lies in the fact that the insects write their spatial memories in the environment, while the mammalian cognitive map lies inside the brain. In his paper entitled 'Cognitive maps in rats and men', Tolman (1948) outlined the evidence upon which he based his theory that rats use field maps of their environment in getting from one place to another. The relevant study (Tolman, Ritchie, and Kalish 1946) used the so-called sun-burst maze.

> *Definitions of the cognitive map range from the general, 'a record in the central nervous system of macroscopic geometric relations among surfaces in the environment used to plan movements through the environment.' (Gallistel, 1990) to the specific, 'The essence of such a structure is the existence of a global representation of objects within some manifold or coordinate system from which their mutual spatial relationships can be derived.'*

As people act in the environment, they perceive surrounding space and acquire knowledge about it. Downs and Stea (1973) called this fundamental process cognitive mapping.

> *Cognitive mapping is formally defined in Downs and Stea (1973) as "... a process composed of a series of psychological transformations by which an individual acquires, codes, stores, recalls, and decodes information about the relative locations and attributes of phenomena in their everyday spatial environment."*
>
> *In other words, Downs and Stea (1973) define Cognitive mapping as the process of acquiring, forming, and maintaining spatial information and spatial knowledge.*

Knowledge acquired during cognitive mapping includes the identities of places and landmarks, the patterns of path connections between places, distances, and directions between places, and so on.

Human Spatial Cognition

The longest standing model of spatial knowledge representation is the Landmark, Route, Survey (or **LRS**) model described by Seigel and White (1975) and Goldin and Thorndyke (1982). This model not only addresses spatial knowledge but also the development process. They have identified three stages of development of an individual's cognitive representation of a large-scale navigable space (Seigel & White, 1975).

- During an initial period of familiarization, a person focuses on the important locations in the environment. Knowledge in this stage consists of a disconnected set of landmarks.
- After more exposure to an environment, people are able to link together important landmarks into routes. Knowledge of this type is said to be a route representation.
- With additional exposure, some people may develop a more flexible, map-like representation of the environment called a survey representation (also known as configurational knowledge).

An individual with a survey representation of a space understands the spatial relationship between the various landmarks in an environment independently of the routes that connect these landmarks. Survey representations facilitate spatial inferences and can allow people access to spatial information regardless of orientation (Sholl, 1987); however, they differ from real environments in well-documented ways (e.g. Tversky, 1981; McNamara, Hardy, & Hirtle 1989; Engebretson & Huttenlocher, 1996). Nevertheless, it is generally assumed that survey knowledge represents a more thorough and flexible understanding of the spatial characteristics of a large scale environment than does route knowledge.
A second model of spatial knowledge is similar to the LRS model but it is hierarchical (Stevens & Coupe, 1978; Colle & Reid, 1998). In some cases, direct exposure to an environment for extremely long durations never results in survey knowledge (Chase, 1983). In other cases, survey knowledge

develops almost immediately. The model proposed by Colle and Reid suggests a dual-mode whereby survey knowledge can be acquired quickly for local regions and slowly for remote regions. The “room effect” comes from the ability to develop survey knowledge of a room rather quickly but survey knowledge of a series of rooms develops relatively slowly and with more errors.

Cognitive Map and Navigation

Navigation and mental maps are intimately tied (Weston and Handy, 2004). The literature strongly indicates that there is a relationship between cognitive maps (CMs), **wayfinding** and navigation. Cognitive maps are used to solve spatial problems and Kitchin suggests that wayfinding and navigation are the most essential spatial problems (Kitchin, 1996). Navigation and wayfinding terms have been used interchangeably to indicate "a person's abilities, both cognitive and behavioral, to reach spatial destinations" (Passini, 1984).

> *Cutting defines wayfinding as a "task to find a way through cluttered environments with ease and without injury"* (Cutting, 1996).

Wayfinding is the cognitive element of navigation. It does not involve movement of any kind but only the tactical and strategic parts that guide movement (Darken, & Allard, 1999).

> *Motion is the motoric element of navigation.*

A reasonable synonym for motion is travel as used by Bowman, et al. (1997). The motor component of navigation refers to the actual locomotion involved.

> *Navigation in physical space (or active navigation) consisting of a cognitive component, often referred to as wayfinding, and a motor component, which is physical locomotion* (Darken, & Allard, 1999).

So navigation is the aggregate task of wayfinding and motion. It inherently must have both the cognitive element (wayfinding), and the motoric element (motion).

> *Locomotion is behavior or movement from one point to another that is guided by one of the senses, most typically vision.*

Wayfinding and Navigation

Wayfinding defined as the mental processes involved in determining a route between two points and then following that route, has long been an important site for studying spatial cognition. Kevin Lynch’s (1960) book - *Image of the City* paid particular attention to making cities more navigable. In a dissertation aimed at modeling common-sense reasoning in general, Benjamin Kuipers chose learning the geography of a place as a case study. The main results were published in a journal article already mentioned, that was to have considerable influence on the field (Kuipers 1978). Shortly after the publication of that work, Riesbeck (1980) described a related problem and implemented a system to judge the clarity of driving directions, given no knowledge of the actual geographical layout.

In the 1980s, the development of microcomputers made it possible to consider designing navigation aid systems for private automobiles that would keep track of the location of the vehicle, relate that position to an on-board digital street map, and provide navigation assistance to the driver. An obvious way to communicate with the driver would be to display maps, and this was the design of early implemented systems, such as Etak’s Navigator (Zavoli *et al.* 1985). A parallel line of work developed systems to provide verbal descriptions of routes, mainly to give to someone renting an automobile and requiring direction to some attraction. Elliott and Lesk (1982), Streeter *et al.* (1985), and Streeter and Vitello (1986) studied the nature and content of driving directions, and related these characteristics to principles of knowledge representation in artificial intelligence. This line of work was picked up by people in the GIS community (Mark 1985, Mark and McGranaghan 1986, Mark *et al.* 1987, McGranaghan *et al.* 1987), and by others working on wayfinding and navigation (Gopal *et al.* 1989, Golledge *et al.* 1993). By the late 1980s, this thread was being related to other aspects of cognitive studies of geographical space and process.

Virtual World Navigation

One of the most complicated tasks when working with virtual worlds is the navigation process. All of us do form and use cognitive maps, whether in real or virtual world, to deal with and process the information contained in the surrounding environment. Cognitive map helps in visualizing the positional and location details and also the route map for reaching the destination from the current location. It helps us find our way in virtual or real environments that we have visited before, and also helps us remember the structure of the place, for example, if we are asked for directions. Quality of such visualizations directly depends on the quality of the cognitive maps. Thus a human being's spatial behavior relies upon, and is determined by the individual's cognitive map of the surrounding environment. Knowledge based systems helps to enhance capacity of machine or computer system to intelligence similar to human being at least in some aspects. Machine based training simulators are equivalent or better than human trainers in terms of efficiency and effectiveness. Our Non-visual spatial navigation (NSL) model falls in this category.

This chapter proposes a typology of navigation strategies and focuses the use of neural networks as a tool for studying navigation within virtual worlds. How it learn to predict the next step for a given trajectory, acquiring also basic spatial knowledge in terms of landmarks and configuration of spatial layout. In addition, how it build a spatial representation of the virtual world, rather like a cognitive map. The benefits of this approach and the possibility of extending the methodology to the study of navigation in Human Computer Interaction and other applications are described in brief. The study of computation models of navigation and the potential of using cognitive maps in the modeling of navigational processes are described. Non-visual spatial learning model is presented for the spatial learning through virtual world exploration. The goal of this model is to provide abstraction of non-visual spatial learning process. Using that assistive system can be developed to facilitate non-visual spatial learning and thereby enhancing mobility skills of person with limited visual. Different types of locomotion in virtual world with their constraints and benefits are discussed. Intended readers of the chapter would be the mobility trainer, architect, city planner, cartographer, psychologist, and game developer.

BACKGROUND - SPATIAL NAVIGATION MODELS

Spatial navigation is an important task for autonomous agents - including humans, animals and robots. A good number of researchers have been working on building computational models of spatial navigation. Goal finding, shortcutting, detouring, exploration and cognitive mapping are just a few of the navigational tasks that are used to test models of navigation. These models take different approaches in describing how individuals perform important navigational tasks.

One approach is to build models that do not take into account the brain regions involved in spatial navigation (Kuipers 1978, Cartwright and Collett 1987, Cheng 1989, Levenick 1991, Chown et al. 1995, Trullier et al. 1997, Reid and Staddon 1998, Voicu and Schmajuk 2001, 2002). These models produce real time behavior that is comparable with individual performance in a given task.

Another biologically inspired approach is to create models that take into account the architecture of the brain regions involved in navigation without aiming to reproduce neuronal activity (Schmajuk and Thieme 1992, Bachelder and Waxman 1994, Benhamou et al. 1995, Samsonovich and McNaughton 1997). These models also produce real time behavior that is comparable with individual performance.

Still another approach is to create computational models of the brain regions involved in navigation that reproduce the experimental findings related to neuronal activity in the brain (Wilkie and Palfrey 1987, Burgess et al. 1994, Arleo and Gerstner 2000, Hasselmo et al. 2002). Owing to the high level of detail in reproducing the parts of the brain involved in navigation, the behavioral analysis includes simple navigational tasks. Freeman (1975) proposed a mathematical model that captures the dynamics

of the cell assemblies based on brain data measured in the olfactory bulb. The impulse response was used to identify the parameters of a second-order linear differential equation that describes the behavior of neuronal populations. Based on this simple model, which simulates the dynamics of cell assemblies, Freeman proposed a hierarchy of models (K0, KI, KII and KIII) that have the capacity to show a periodic behavior similar to that found in the EEG (Freeman et al. 1997) and can be used to perform robust pattern recognition of noisy and variable data (Kozma and Freeman 2001).
Here in this section we briefly discuss the following **spatial learning models**.

- Symbolic model (Kuipers, 1978)
- Connectionist model (Schmajuk and Thieme, 1992)
- Attractor networks model (Samsonovitch and McNaughton, 1997)
- Biologically inspired chaotic neural network model (Kozma et al., 2003)
- Neural network model incorporating a hierarchical cognitive map (Voicu, 2003)
- A Multiscale progressive model (Zhang, 2008)

Symbolic model (Kuipers, 1978)

Kuipers presented a model of the knowledge a person has about the spatial structure of a large-scale environment: the "cognitive map." People navigate by using their common-sense knowledge of space: they link many separate observations into a "cognitive map" and solve route-finding problems by consulting the knowledge in the cognitive map. A large-scale space is defined by the way it is perceived, not by its physical size. Its structure is deduced from a number of observations over time, rather than being perceived from one location. The places, paths and routes of a city are the most common large-scale space in most people's experience, although of course there are others: ocean, jungle, and open field.
The term "cognitive map" as used in his model, refers to a body of knowledge that a person has in his head about the spatial structure of an environment. This is to distinguish it from other related concepts, such as the expression of that knowledge as maps, models, or verbal directions. People also associate visual impressions or social facts with different parts of the environment, but those are not the structural aspects of the cognitive map that will concern us here.
The functions of the cognitive map are to assimilate new information about the environment, to represent the current position, and to answer route-finding and relative-position problems. This model (called the TOUR model) analyzes the cognitive map in terms of symbolic descriptions of the environment and operations on those descriptions.
Knowledge about a particular environment is represented in terms of route descriptions, a topological network of paths and places, multiple frames of reference for relative positions, dividing boundaries, and a structure of containing regions. The current position is described by the ***"You Are Here"*** pointer, which acts as a working memory and a focus of attention. Operations on the cognitive map are performed by inference rules which act to transfer information among different descriptions and the "You Are Here" pointer. The TOUR model shows how the particular descriptions chosen to represent spatial knowledge support assimilation of new information from local observations into the cognitive map, and how the cognitive map solves route-finding and relative-position problems.
A central theme of this research is that the states of partial knowledge supported by a representation are responsible for its ability to function with limited information or computational resources. The representations in the TOUR model provide a rich collection of states of partial knowledge and, therefore, exhibit flexible, "common-sense" behavior.

Connectionist (Schmajuk and Thieme, 1992)

Schmajuk and Thieme (1992) have offered a biologically plausible theory that includes (a) an action system consisting of a goal-seeking mechanism with goals set by a motivational system and (b) a cognitive system in which a neural network implements a cognitive map. The cognitive map represents the connectivity between places and the connectivity between places and goals. The goal-seeking mechanism changes from random exploratory behaviour to approach behaviour when either (a) the goal is found or (b) one place in the cognitive map generates a prediction of the goal that is stronger than the predictions generated by all other alternative places.
Voicu and Schmajuk (2001a, b) described a modified version of the Schmajuk and Thieme (1992) model. The new model differed from the original one in two ways. First, whereas the early model assumed no a-priori knowledge of the space to be explored, the modified model assumed a representation of the environment as a set of potentially connected and unexamined locations. Second, whereas in the original model the decision of what place to move next was based on the comparison of the predictions of the goal when each of the alternative places is briefly entered, in the new model this decision was based on the comparison of the activation of each of the alternative places when the goal is activated.
Fig. 1 shows the Schmajuk and Thieme (1992) model of spatial exploration and navigation with the modifications introduced by Voicu and Schmajuk (2001a, b). The system entails (a) an action system including a goal-seeking mechanism with goals defined by (b) a motivation system, (c) a cognitive system including a neural network which implements a cognitive map, and (d) a short-term memory where the reading of the cognitive map is temporarily stored.

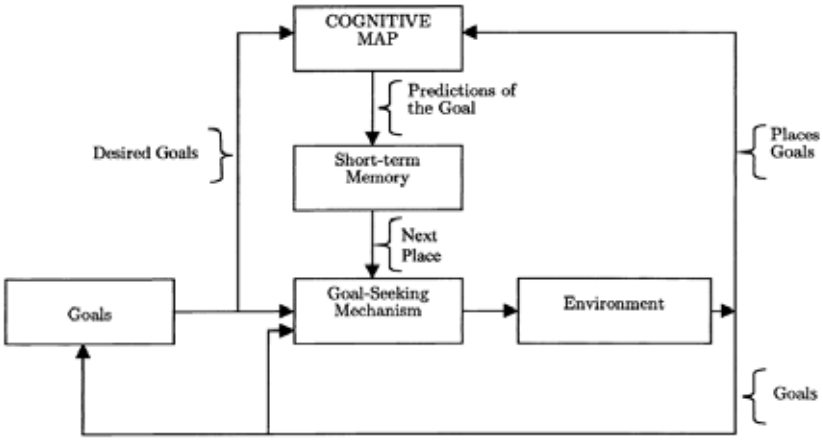


Fig. 1. A system for spatial exploration and navigation. Block diagram of the model showing the interaction between the action system (goal-seeking mechanism), motivation system (goal), cognitive system (cognitive map), short-term memory, and environment.

Once the motivation system defines a goal for the simulated animal (e.g. food for hunger, unexamined places for exploration), the action system starts the search for those goals. If the goal is perceived, the simulated animal moves towards it. If it is not perceived, but it can be predicted by the cognitive map, the simulated animal enters the place that best leads to the location of the goal. If the goal is neither perceived nor predicted, then the simulated animal engages in exploratory behaviour using the cognitive map.

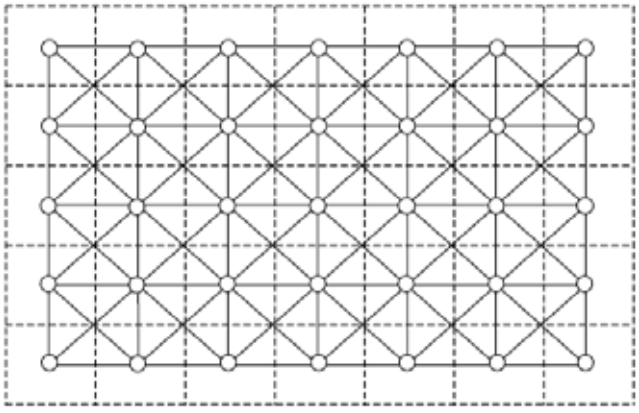


Fig. 2. The canvas. Squares in broken lines represent the places to be explored. Solid lines represent connections between places. The empty canvas is a lattice representing the potential continuity of the space to be explored. Adjacent places are assumed to be linked and each place is designated as unexamined (represented by an open circle.)

The map of the environment is drawn on an empty canvas that represents each location in the space to be mapped. Fig. 2 shows that the canvas is a lattice representing the potential continuity of space. In the figure, dashed-side squares represent places, the circles indicate their centres, and the solid lines linking the circles represent possible movements from one place to another. Although the places used here are square, places with arbitrary forms can also be used as long as they preserve the continuity of space. Places are of approximately the size of the footprint of the agent.

Attractor networks (Samsonovitch and McNaughton, 1997)

A minimal synaptic architecture was proposed for how the brain might perform path integration by computing the next internal representation of self-location from the current representation and from the perceived velocity of motion. In the model, a place-cell assembly called a "chart" contains a two-dimensional attractor set called an "attractor map" that can be used to represent coordinates in any arbitrary environment, once associative binding has occurred between chart locations and sensory inputs. In hippocampus, there are different spatial relations among place fields in different environments and behavioral contexts. Thus, the same units may participate in many charts, and it is shown that the number of uncorrelated charts that can be encoded in the same recurrent network is potentially quite large. According to this theory, the firing of a given place cell is primarily a cooperative effect of the activity of its neighbors on the currently active chart. Therefore, it is not particularly useful to think of place cells as encoding any particular external object or event. Because of its recurrent connections, hippocampal field CA3 was proposed as a possible location for this "multichart" architecture; however, other implementations in anatomy would not invalidate the main concepts. The model is implemented numerically both as a network of integrate-and-fire units and as a "macroscopic" (with respect to the space of states) description of the system, based on a continuous approximation defined by a system of stochastic differential equations. It provides an explanation for a number of hitherto perplexing observations on hippocampal place fields, including doubling, vanishing, reshaping in distorted environments, acquiring directionality in a two-goal shuttling task, rapid formation in a novel environment, and slow rotation after disorientation. The model made several

new predictions about the expected properties of hippocampal place cells and other cells of the proposed network.

Biologically inspired Chaotic Neural Network Model (Kozma et al., 2003)

K model is a biologically plausible neural network, the development of which was based on the salamander's central nervous system. The model has the capacity to show a periodic and chaotic behavior similar to those found in the animal EEG patterns. KIII can be used to perform robust pattern recognition of noisy and variable data. The family of K sets includes K0, KI, KII, KIII and the highest level, KIV. K0 is a single node with nonlinear input-output transfer features. KI is a coupling of either excitatory or inhibitory K0 units, while KII is a double layer of excitatory and inhibitory units. KIII is a set of two or more KII units connected by feed forward and delayed feedback connection. With the proper parameters selection, the model can maintain non-convergent chaotic oscillation among all the excitatory and inhibitory nodes in the system.
The discovery that brain dynamics exhibits chaotic features has profound implications for the study of higher brain function (Skarda, C.A. & Freeman, W.J., 1987 and Schiff, S.J., 1994). The KIII model is a working example of the implementation of these chaotic principles in a computer environment. KIII exhibits a number of experimentally observed behaviours of brains, like robust pattern recognition and classification of input stimuli, and fast transitions between brain states (Chang H.J. & Freeman W.J., 1996, Freeman, W.J., 2000, Kozma, R., and Freeman, W.J., 2001, Kozma, R., et al., 2001). KIII shows very good performance in the several kinds of learning needed to categorize input data, and it can generalize efficiently in various classification problems. The operation of the KIII model can be described as follows. In the absence of stimuli the system is in a high dimensional state of spatially coherent basal activity, which is described by an aperiodic (chaotic) global attractor. In response to external stimuli, the system can be kicked-off the basal state into a local memory wing. The system resides in the localized wing for some time, then it returns to the basal state. This is a temporal burst process of the duration of up to 200 milliseconds. See Viana Di Prisco, G & Freeman, W.J., 1985 for the olfactory bulb, and Barrie J.M., Freeman W.J. & Lenhart M.D., 1996 for neocortex.
The next highest level of the K-sets is the KIV model. As in the case of all other K-sets (Freeman, 1975), the architecture and functionality of KIV is strongly biologically motivated. The data that are required for modeling neurodynamics at this level are derived by analysis of EEG recorded simultaneously from multiple electrodes in sensory and limbic structures (Kay, L.M & Freeman, W.J., 1998). KIV provides the neuroarchitecture that is needed to model the interactions of key neural populations in the primordial vertebrate forebrain. Among these parts are the sensory cortices, the motor cortices and nuclei, and the hippocampal formation, which is essential for cognitive processes such as orientation, learning and memory. Three types of sensory signals are considered in KIV: exteroceptors, interoceptors (including proprioception), and orientation signals; e.g., gravity, visual flow, magnetic fields. Each of these sensory signals provide stimuli towards the brain, namely the sensory cortices, midline forebrain (MF) unit, and the hippocampal formation (HF), respectively; see Fig. 3. The present model is motivated by the architecture and putative function at the level of the amphibian brain. It is not intended to mimic all the biological details; rather it is used to incorporate the main elements required for operation of brains at the KIV-level of functionality.

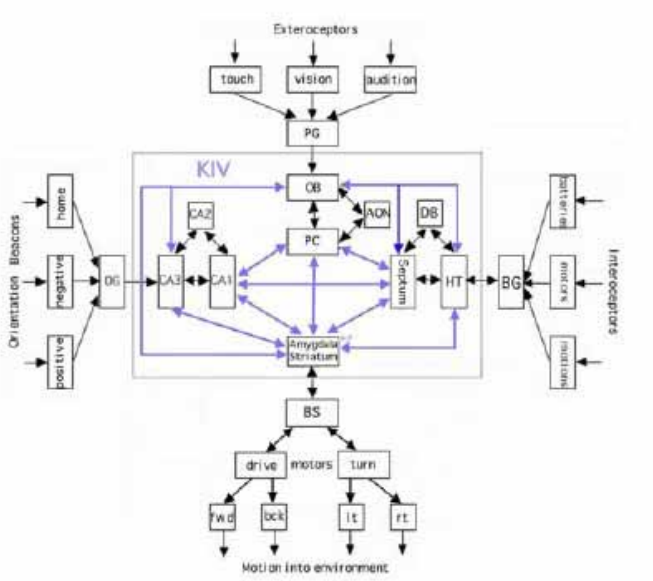


Figure 3. Structure of the KIV model based on Kozma, Freeman, and Erdi (2003). Abbreviations: DG, dentate gyrus; CA1-CA3, Cornu Ammonis (hippocampal sections); PG, periglomerular; OB, olfactory bulb; AON, anterior olfactory nucleus; PC, prepyriform cortex; Spt, septum; DB, diagonal band; HT, hypothalamus; BC, basal ganglia, corpus striatum including thalamus; BS, brain stem. The sparse long connections that comprise the KIV set are shown as bi-directional, but they are not reciprocal. The entorhinal cortex is omitted, because that is a neocortical structure found only in mammals.

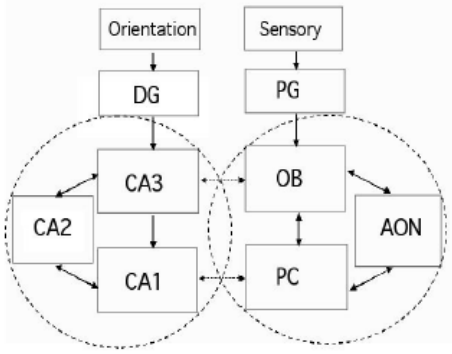


Figure 4. Schematic view of the simplified KIV model with the interacting hippocampal and cortical KIII units.

The hippocampus is the main focus of cognitive mapping that supports spatial navigation and temporal orientation (short term memory). There is a rich literature of hippocampal-based navigation models (Burgess, N., Recce, M., & O'Keefe, J., 1994, and Arleo, A. & Gerstner, W., 2000). In their model, the following parts of the hippocampus are modeled: Dentate Gyrus (DG), CA3, CA1, and CA2. They need CA2 in our model to generate the hippocampal KIII dynamical system, serving as its chaotic controller. CA1, CA2 and CA3 are modeled as KII units, while DG will be a KI unit. KII units are shown to generate point attractors, limit cycle attractors, and even chaotic attractors (though lacking robustness) in the gamma band.

They model the sensory cortical (SC), midline forebrain (MF), and the hippocampal formation (HF) systems as KIII sets. Each KIII set has three KII units as components and exhibits robust aperiodic oscillations in the gamma range. Each shows spatial coherence in the form of a shared, spatially distributed, aperiodic wave form, with amplitude modulation patterns occurring in sequential frames over time, indicating the existence of landscapes of chaotic attractors corresponding to categories of sensory stimuli that have been learned. These AM patterns are manifestations of self-organizing dynamics that creates coherent activity in the form of "wave packets" (Freeman, W.J., 1975) as vectors of information in perception. The gating of bursts of KIII activity is governed by a limit cycle attractor in the KII set modeling the septum, that is fixed at frequency in the theta band, here 5 Hz (analogous to a sniff or saccade).
The cortical KIII system initiates the function of pattern recognition by the agency of sensory input-induced destabilization of high-dimensional dynamics. This actualizes an attractor landscape formed by previous experience in the OB/PC, which in our model is the common sensorium for all distance receptors, as it is in the salamander (Herrick, C.J., 1948). The hippocampal KIII system, thereafter, uses the categorization embodied in the outputs of the OB and PC as its content-laden input, to which the DG contributes the temporal and spatial location of the environmental events.
Another KIII component of the integrated KIV system, the Midline Forebrain formation, receives the interoceptor signals through the basal ganglia, and processes them in the hypothalamus and the septum. MF provides the value system of the KIV, using information on the internal goals and conditions in the animal. It provides the “Why?” stream to the amygdala, which combines this with the “What?” and “Where?” information coming from the cortex and the hippocampus to make a decision about the next step/action to be taken.
The motor part of the model limbic system is driven by the simulated amygdala. The direction of motion that it determines is based on the combined information from the three sensory systems, which collectively form the architecture of the global KIV. From EEG studies we infer that a cooperative state emerges from the collective interaction among the CA1, PC, Septum, and Amygdala, by which various behavioural patterns are formed and executed. The model given in Fig. 1 is designed to provide the platform with which to study by simulation this behaviour formation and action selection mechanism. In a very simple approach, however, we define initially only three basic behaviours: wall following, object avoidance, and backup. Backup behaviour is invoked if the robot is stuck or cannot execute a chosen action. A wide range of problems of intentional action can be solved with these three simple behaviours.

Neural network model incorporating a hierarchical cognitive map (Horatiu Voicu, 2003)

They describe a computational model of spatial navigation based on experimental studies conducted with human participants. The model builds and uses a hierarchical cognitive map of a large environment. Computer simulations show that the model correctly describes experimental results including hierarchical organization of space and distance estimation. Furthermore, the model predicts that reaction time for distance estimation varies nonlinearly with distance.
The architecture of the model, except the hierarchical cognitive map, is based on a previous model (Voicu & Schmajuk, 2001). It contains five modules: a localization system that provides a unique code for each landmark in the environment, a cognitive map that builds and updates a hierarchical representation of space, a working memory used for path planning, a motor system that translates the path planning information into motor action and a control system that supervises the flow of information between the above modules (Fig. 5).

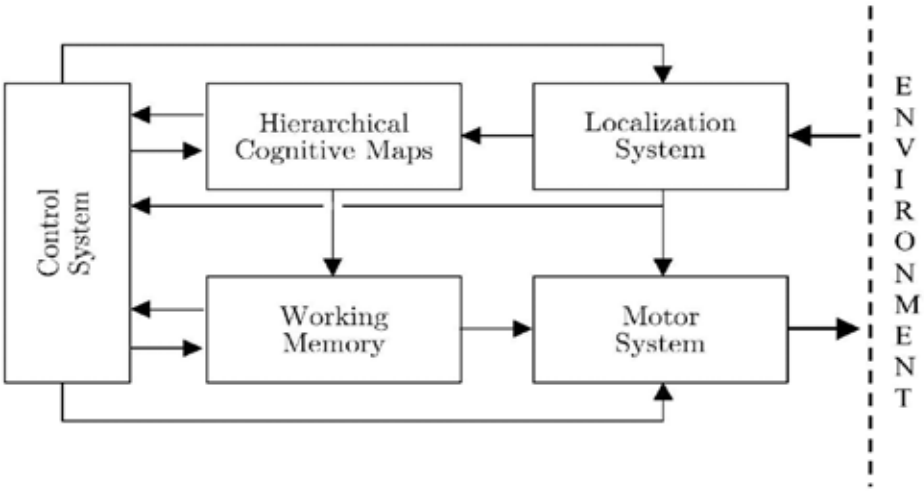


Fig. 5. Block diagram of the model. It shows the interaction between the control system, localization system, cognitive map, working memory, motor system and environment.

The localization system recognizes landmarks that are contained in the sector of circle that has the center in the position of the agent, an angle of 608 and a radius equal to one-tenth of the length of the environment. The orientation of the agent is given by the bisecting line of the sector of circle. The representation of the landmarks recognized by the localization system are stored in a short-term memory and provided to the cognitive map. The working memory stores information retrieved from the cognitive map and used by the motor system to plan and execute paths in the environment. The cognitive map contains three associative memories, implemented by hetero-associative networks (Kohonen, 1977), that build association between landmarks. All networks have recurrent connections and can exchange information. Each associative memory stores spatial information about the environment at a different resolution. The first level stores associations between all the landmarks in the environment. The second level stores associations between landmarks that have the largest number of associations at the first level. The third level stores associations between landmarks that have the largest number of associations at the second level. This hierarchy is acquired while the agent explores the environment. The cognitive map contains no a priori information related to spatial organization.

A Multiscale Progressive Model (Xiaolong Zhang, 2007)

This model sees exploration-oriented navigation as an evolving process in which navigation tasks are gradually refined. Consider a scenario of going to a conference in an unfamiliar city. We usually decompose the overall goal, which is to get to the conference room, into a sequence of sub-goals: going to the neighborhood of the conference place, finding the conference place, and locating the conference room. The overall navigation task is a set of subtasks at different levels. Each subtask involves such activities as sub-goal planning, moving, perceiving the environment, and assessing the movement results (Figure 6a). The accomplishment of a subtask is the condition of the pursuit of the subtask at the next level (Figure 6b).

Completing a subtask requires spatial knowledge. We need spatial knowledge for goal planning, movement guidance, and navigation result assessment. During moving, we perceive the environment and update our spatial knowledge. If the sub-goal has not been achieved, further movement would be needed. Otherwise, we shift navigation to the subtask at the next level. For a particular subtask, we will need spatial knowledge appropriate to this task. When we complete one subtask and move to the

subtask at another level, required spatial knowledge also changes. For example, to find the conference place, we need a map of its neighborhood, but after we have located the conference place, we will need a floor map to help us find the conference room. What has been changed with the level of subtasks also includes action accuracy. Different subtasks may have different requirements for movement accuracy. Being in the neighborhood of the conference place needs only be accurate to tens or even hundreds of meters, but finding the door to the conference room requires position accuracy at the level of the meter. Different movement accuracies imply that people can choose different movement speeds for different subtasks to achieve efficiency, as long as movement accuracy can satisfy the error tolerance for each subtask.

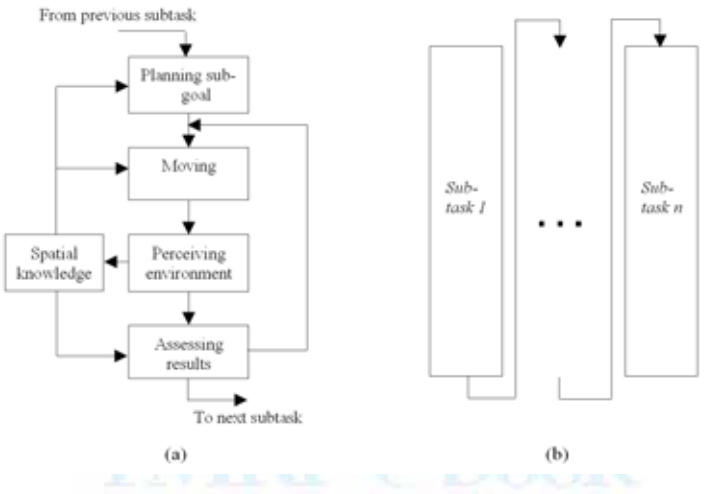


Figure 6. A multiscale progressive model on navigation: a) is the subtask unit that shows the activities needed to complete a subtask; b) indicates the decomposition of the overall navigation task into a set of subtasks, each of which is a subtask unit.

Spatial knowledge and movement at each level need to be comparable. Spatial knowledge guides movement, and movement updates spatial knowledge. Unmatched spatial knowledge and movement may make it difficult for people to plan their actions and assess the results of the actions. For example, a map of the conference place area with a resolution of tens or hundreds of meters would be suitable for locating the conference place, but not for finding the conference room, which requires a floor map of the conference place with a resolution at the meter level. Such cross-scale transition should be smooth so that users can easily see how objects at different scales are related to each other. This will allow users to establish the connection between objects and to align spatial structures easily. For example, in order to be a specific place of a building, a user can first choose a scale level to approach the target building quickly, and then change to another scale level to adjust the final location accurately. A smooth transition of spatial knowledge between these two different scales makes the targeted building always visible. The object constancy in the smooth transition provides the user with the frame of reference in navigation. The user can clearly see how new spatial information is related to previous one and easily know where to go next for the following subtask. The granularity transformation model of wayfinding by Timpf and Kuhn (2003) saw wayfinding in the real world as a hierarchical process based on the differently-scaled maps. Their model largely focused on the cognitive aspect of navigation activities in goal planning, strategy choosing, and moving. The progressive model proposed by him, by considering both spatial knowledge and spatial actions-movement, emphasizes

that navigation is an evolving process with gradually refined sub-goals, and argues the necessity of coupling spatial knowledge and movement as well as easily transferring them across different sub goals.

ARTIFICIAL INTELLIGENCE IN GEOGRAPHICAL CONTEXTS

As the field of artificial intelligence matured, the knowledge representation and processing methods developed in the laboratory were applied to real world problems in the 1970s. Several applications to geographical information became visible. In 1978, Benjamin Kuipers published an elaborate model for representing human knowledge of large-scale space in an artificial intelligence framework (Kuipers, 1978). Kuipers' model built largely on the intuitive and descriptive work of Kevin Lynch (1960). In a follow-up paper, Kuipers discusses the `map in the head' metaphor frequently employed to account for people's ability to find their way in space (Kuipers, 1982). The author uses computational models to refine the too simplistic metaphor and to discuss its implications in detail. In a subsequent paper, Kuipers discusses cognitive maps, their structure, and potential alternatives by employing a thought experiment in robotics (Kuipers, 1983). Shortly after the publication of that work, Riesbeck (1980) described a related problem and implemented a system to judge the clarity of driving directions, given no knowledge of the actual geographical layout. Davis (1983, 1986) looked at a cognitive map as a knowledge base; he developed a theory of representation, retrieval, and assimilation of geographical knowledge and implemented his theory in the MERCATOR system. MERCATOR is conceived for the use by a robot whose task is to build up a coherent representation of his visually perceived environment. Yeap (1988) also developed a computational theory of cognitive maps. Yeap's work emphasizes the cooperation of different loosely coupled modules representing different levels of information. The approach is motivated by Marr's (1982) investigations into the human representation and processing of visual information. Finally, Munro and Hirtle (1989) and Wender (1989) have developed connectionist models of cognitive maps, in contrast to the symbolic approaches listed above. Critical evidence from linguistics entered the picture in 1983 with Leonard Talmy's seminal work on how space is structured in language (Talmy 1983). This paper started the important cognitive linguistic research thread in cognitive geographical research. The artificial intelligence system CITYTOUR (Andre *et al.* 1987) was designed to answer natural language questions about the spatial relationships between objects in a city. Approaches from artificial intelligence were also applied to classic problems in cartography, especially automated name placement and other aspects of map design (Freeman and Ahn 1984, Buttenfield and Mark 1990). Couclelis (1986) reviewed artificial intelligence in geography during its early stage of widespread impact in the discipline.

NON-VISUAL SPATIAL LEARNING (NSL) MODEL

Although a good number of researchers worked on **non-visual spatial learning**, to the best of our knowledge, yet no researcher has given a computation model to cover all the aspects of the non-visual spatial learning. Understanding how spatial learning tasks are constructed is useful in determining how best to improve navigation performance. We should decompose the various tasks of spatial learning in a generic way. So that we might be able to determine where assistance is needed, or where more learning can occur.

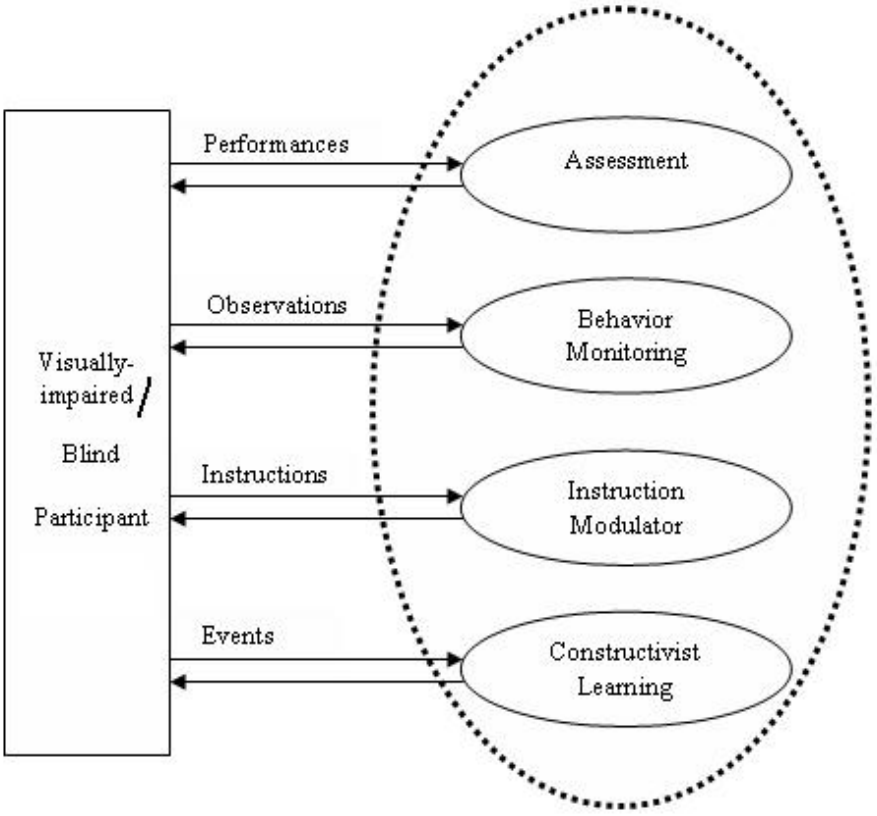


Fig.7. Components of Non-visual Spatial Learning (NSL) Model

To describe the computational model of the spatial learning, we divided whole process into following four steps (see Fig. 7).

1. Constructivist Learning
2. Instructor modulator
3. Behavior monitoring
4. Assessment

The model works like this. In first phase, a **constructivist learning** experience is to be provided that emphasizes the active participation of users in virtual environment exploration. Virtual environment exploration should be as familiar and natural as actually walking through the regions portrayed on a traditional paper map. In case of confusion or any difficulty, user can take assistance from it.
In second phase, simulated agent explores the area and creates the knowledgebase to generate the various kinds of instructions for providing assistance. Instructor modulator conforms the instructions and conveys to participants in various ways (speech, force feedback, and/or non-speech sound). It also provides guidance and directs the participant. Besides this assistance, participant can get contextual cues that help them to structure cognitive maps. Participant can interact with the various objects of virtual environment and structure cognitive map of an environment.

In third phase, partial cognitive map build till now, it is evaluated in terms of participant's behavior, navigating style (i.e. normal walk or drunkard/random walk) and participant's course with obstacles (where and when). Need based further instructions may be provided for any adjustment.
In the final phase, once the participant gets confident and memorizes the path and landmarks between source and destination, he is allowed to go for assessment or is confined to training. Participant's navigation performance, such as path traversed, time taken and number of steps taken to complete the task are recorded and evaluated. The participant's performance is evaluated in terms of statistical measures like bi-dimensional correlation coefficients (BCC), navigation time, distance traveled and number of steps taken to reach the destination place from source place. The sequence of objects falling on the traversed path and the positions where he seemed to have confusion (and hence took relatively longer time) are also recorded and conveyed to them. Performance feedback is to be given to participant.

Constructivist Learning

A constructivist learning experience is to be provided that emphasizes the active participation of users in spatial learning through virtual environment exploration. This is kind of learning-by-exploring approach of learning. The participant interacts with the various objects of an environment. Also virtual environment exploration should be as familiar and natural as actually walking through the regions portrayed on a traditional paper map.
Participant can load layout of premises or areas through interface easily. Participant can get the brief description of the layout in the beginning. Participant can choose starting location and destination through text-to-speech guided selection. Participant can start Session by pressing key (for say F5). Session start date-time (that is also start break date-time for first break) is to be stored in the database. Now participant can start navigation. The participant navigates or explores the virtual environment using a force feedback joystick, mouse or locomotion interface. Before starting training session, participant can configure a) foot step size, b) height and c) length of their foot. Participant can start the session by pressing a particular key. System maintains session starting and ending time. Participant can end the session by pressing a particular key. In case of confusion or any difficulty, participant can take assistance from it. Participant can get information regarding orientation, whenever needed, through help key (for say F3). Participant can also get information (help) regarding Near-by (, near to near-by, or far) objects from his current location (i.e. knowing orientation) by pressing key (for say F4). Participant also gets options/direction available to move (from shortest path) by pressing key (for say F8), number of steps taken and distance covered from origin or from particular location. Participant also gets information like distance remains to travel to reach the destinations. When participants takes these type helps, system stores information regarding helps taken (i.e. When and Where – Current location). This information is used to find the confidence level of the participants. The System also generates vibration and audible alert when the participant is approaching any obstacle.
Following are the some of the common operations need to be implemented:

```
Load layout (as prepared by sighted)                    by participant
        void loadLayout(layout_name);
Select source and destination locations                 by participant
        void selectSourceLocation(location_name);
        void selectDestinationLocation(location_name);
Starting and ending session                              by participant
        int startSession();
        void endSession(sessionNumber);
Take break                                              by participant
```

```
        int takeBreak();      return next break number.
Transition / Acceleration                                   by system
        void transition (stepNumber);
        void accelerate();
Get orientation help                                        by participant
        String[] getOrientationHelp();
Get present route help                                      by participant
        String getPresentRouteHelp();
Taking assistance                                           by participant
        void askOptionToMove(presentPosition);
```

Instruction Modulator

This is kind of directed mode of navigation. The basic concept of directed mode of navigation is to augment the standard user interface in a virtual environment with a system that has knowledge about the content of the virtual world and lets users find objects and locations through assistance.
Following are the type of instructions provided by the system:

1. Directional Instructions
2. Obstacles avoidance Instructions
3. Orientation Instructions
4. Contextual Instructions
5. Temporal Instructions

In this mode of navigation, the **Instruction modulator** guides the participant through speech by describing surroundings, guiding directions, and giving early information of a turning, crossings, etc. Additionally, occurrences of various events (e.g. arrival of a junction, arrival of object(s) of interest, etc.) are signaled through vibration using consumer-grade devices. Current position in virtual environment is changed as user walks on the locomotion interface. They may use the force feedback joystick, mouse to control and move the current position indicator (referred to as cursor in this chapter). System generates (non-speech) sound for each step taken by User. Whenever the cursor is moved onto or near an object, its sound and force feedback features are activated. Thus a particular sound, which may also be a pre-recorded message, will be heard by the participant. As long as the cursor is on the object, the participant will feel the force feedback effect associated with this object.
Participant can get contextual information continuously during navigation according to session's time like Morning, Afternoon, Evening, or Night. Contextual information is also according to different events of the place. For example for Railway station, contextual information is different for different events like at train arrival time, departure time and normal waiting time period. When participant is approaching or passing through class room (for school or college premises) he gets sound of teacher and students with Doppler Effect. When participant is passing through fountain or river, they heard the sound of flowing of water. Also participant gets information about path surface (i.e. Sandy, Muddy, Concrete, Slippery, or Grass-root/loan, etc.) through tactile effects.

Behavior monitoring

The system continuously monitors and records following type of participant's behaviors:

1. Navigating style (normal /random walk)
2. Mental state (confusion/fear/lost/excited/confident)

During learning, the system continuously monitors and records participant's navigating style (i.e. normal walk or drunkard/random walk) and participant's course with obstacles (where and when). Participant can take break at any time during the session by pressing the Escape key. System stores break number, break start time and end time, and user's trajectories in the database.
Once the system finds the participant confident and memorizes the path and landmarks between source and destination, it allows him to go for assessment. The system monitors the number of step taken and distance traveled by the participant for the each break and session. If these two values are reducing or coming near the expected values and if participant's navigation style is proper then system finds that the participant is confident and ready for performance test.

Assessment

In the assessment phase, participant navigates without system's help and trying to reach the destination. Participant gets only contextual information. The system records participant's navigation performance, such as path traversed, time taken, distance traveled and number of steps taken to complete this task. It also records the sequence of objects falling on the traversed path and the positions where he seemed to have confusion (and hence took relatively longer time). The system evaluates the participant's performance in terms of statistical measures like bi-dimensional correlation coefficients (BCC), navigation time and number of steps taken by participant to reach the destination place from source place and gives performance feedback to participant.
As mentioned earlier the simulated agent finds out the possible best path(s) between source place and destination using optimal path algorithm. The system compares optimal paths with participant's earlier recorded navigation path (during un-guided virtual navigation). Navigation paths of the participant are evaluated quantitatively using bi-dimensional regression analysis developed by Tobler. Bi-dimensional regression is applied to calculate the bi-dimensional correlation coefficients. The value of BCC near to one indicates that participant's performance is satisfactory and cognitive maps are satisfactorily formed. If its value is one then the cognitive map is almost precise. The trainer may ask the participant to give a verbal description of the area and then performs orientation and mobility tasks in the real target space.

Quality factors for spatial learning techniques

There are few categories of virtual environment applications that are currently in use for productive, consistent work, but the requirements of these applications for spatial learning techniques cover a wide range. Further, there are many new applications of VEs being researched, which also may require spatial learning techniques with different characteristics. It is therefore impractical to evaluate spatial learning techniques directly within each new application. Instead, we propose a more general methodology, involving a mapping from spatial learning techniques to a set of quality factors. Quality factors are measurable characteristics of the performance of a technique. With this indirect mapping, application designers can specify desired levels of various quality factors, and then choose a technique which best fits those requirements. Our current list of quality factors for VR-based spatial learning techniques includes:

- Speed of learning (time taken to develop cognitive map)
- Navigation Efficiency (Number of steps taken and distance traveled to complete the task)
- Accuracy (proximity to the desired target)
- Spatial Awareness (the user's knowledge of his position and orientation within the environment during and after exploration)
- Ease of Learning (the ability of a novice user to use the technique)

- Ease of Use (the complexity or cognitive load of the technique from the user's point of view)
- Information Gathering (the user's ability to actively obtain information from the environment during exploration)
- Presence (the user's sense of immersion or 'being within' the environment due to navigation)
- User Comfort (lack of simulator sickness, dizziness, or nausea)

This list may not be complete, but it is a good starting point for quantifying the effectiveness and performance of virtual spatial learning techniques. Some of the quality factors, such as speed, navigation Efficiency and accuracy, are simple to measure quantitatively. Others, however, are difficult to measure due to their inherent subjective nature. To quantify these factors, standard questionnaires for factors such as ease of use (e.g. Chin, Diehl, & Norman, 1988), presence (e.g. Slater, 1995), and simulator sickness (e.g. Kennedy et al., 1993) should be part of the experimental method.

LOCOMOTION IN VIRTUAL WORLD

Virtual reality provides for creation of simulated objects and events with which people can interact. The definitions of virtual reality (VR), although wide and varied, include a common statement that VR creates the illusion of participation in a synthetic environment rather than going through external observation of such an environment (Earnshaw, R. A., Gigante, M. A., & Jones, H. (1993)). Essentially, virtual reality allows users to interact with a computer-simulated environment. Users can interact with a virtual environment either through the use of standard input devices such as a keyboard and mouse, or through multimodal devices such as a wired glove, the Polhemus boom arm, or else omni-directional treadmill. The **locomotion interface** is used to simulate walking from one location to another location. The device is needed to be of a limited size, allow a user to walk on it and provide a sensation as if he is walking on an unconstrained plane.

Generally, a locomotion interface should cancel the user's self motion in a place to allow the user to go to anywhere in a large virtual space on foot. For example, a treadmill can cancel the user's motion by moving its belt in the opposite direction. Its main advantage is that it does not require a user to wear any kind of devices as required in some other locomotion devices. However, it is difficult to control the belt speed in order to keep the user from falling off. Some treadmills can adjust the belt speed based on the user's motion. There are mainly two challenges in using the treadmills. The first one is the user's stability problem while the second is to sense and change the direction of walking. The belt in a passive treadmill is driven by the backward push generated while walking. This process effectively balances the user and keeps him from falling off.

The problem of changing the walking direction is addressed by Brooks, F. P. Jr., (1986) and Hirose, M. & Yokoyama, K., (1997), who employed a handle to change the walking direction. Iwata, H. & Yoshida, Y., (1997) developed a 2D infinite plate that can be driven in any direction and Darken , R. P., Cockayne, W.R., & Carmein, D., (1997) proposed an Omni directional treadmill using mechanical belt. Noma, H. & Miyasato, T., (1998) used the treadmill which could turn on a platform to change the walking direction. Iwata, H. & Fujji, T., (1996) used a different approach by developing a series of sliding interfaces. The user was required to wear special shoes and a low friction film was put in the middle of shoes. Since the user was supported by a harness or rounded handrail, the foot motion was canceled passively when the user walked. The method using active footpad could simulate various terrains without requiring the user to wear any kind of devices.

Type of locomotion in virtual world

It has often been suggested that the best locomotion mechanism for virtual worlds would be walking, and it is well known that the sense of distance or orientation while walking is much better than while

riding in a vehicle. However, the proprioceptive feedback of walking is not provided in most virtual environments. Good number of devices has been developed over the last two decades to integrate locomotion interfaces with VR environments. We have categorized the most common VR locomotion approaches as follow:

- walking-in-place devices (Sibert, Templeman, Page, Barron, McCune & Denbrook, 2004; Whitten et al, 2008),
- treadmills-style interface (Darken, Cockayne & Carmein, 1997; Hollerbach, Xu, Christensen, & Jacobsen,2000; Iwata, & Yoshida, 1999; De Luca, Mattone, & Giordano, 2007),
- the motion foot pad (Iwata, Yano, Fukushima, & Noma, 2005),
- robot tiles, actuated shoes (Iwata, Yano, & Tomioka, 2006),
- the string walker (Iwata, Yano, & Tomiyoshi, 2007), and
- Finger walking-in-place approach.

CONCLUSION

This model is effective to promote the development and online evaluation of cognitive maps of users. **Knowledge based systems** help to enhance capacity of machine or computer system to behave intelligently, similar to human being in some aspects at least. Machine based training simulators are equivalent or better than human trainers in terms of efficiency and effectiveness. Our Non-visual spatial navigation (NSL) model provides computational framework for spatial knowledge representation, acquisition and assessment of the acquired spatial knowledge. We are encouraged by preliminary results from our prototype implementation, which suggest that such spatial learning techniques would help visually challenged and blind people to get effectively learned for independent navigation. This is an ongoing study and we feel that our system based on our NSL model will be progressively enriched to become increasingly effective for spatial learning by them.

REFERENCES

André, E., Bosch, G., and Rist, T., 1987, Coping with the intrinsic and deictic uses of spatial prepositions. In Artificial Intelligence II, Proceedings of the Second International Conference on Artificial Intelligence: Methodology, Systems, Applications, Varna, Bulgaria, edited by P. Jorrand and V. Sgurev (Amsterdam: North-Holland), pp. 375 - 382.

Arleo, A., and Gerstner, W., 2000, Spatial cognition and neuro-mimetic navigation: a model of hippocampal place cell activity. Biological Cybernetics, 83: 287–299.

Barrie J.M., Freeman W.J., and Lenhart M.D., 1996, Spatiotemporal analysis of prepyriform, visual, auditory, and somesthetic surface EEGs in trained rabbits, J. Neurophysiol., 76: 520-539.

Bachelder, I. A., and Waxman, A. M., 1994, Mobile robot visual mapping and localization: a view based neurocomputational architecture that emulates hippocampal place learning. Neural Networks, 7: 1083–1099.

Benhamou, S., Bovet, P., and Poucet, B., 1995, A model for place navigation in mammals. Journal of Theoretical Biology, 173: 163–178.

Brooks, F. P. Jr., 1986, Walk Through- a Dynamic Graphics System for Simulating Virtual Buildings. Proc. of 1986 Workshop on Interactive 3D Graphics, pp. 9-21.

Bowman, D., Koller, D., and Hodges, L., 1997, Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques. Proceedings of the Virtual Reality Annual International Symposium, 45-52.

Burgess, N., Recce, M., and O'Keefe, J., 1994, A model of hippocampal function. Neural Networks, 7 (6/7): 1065-1081.

Buttenfield, B. P., and Mark, D. M., 1990, Expert systems in cartographic design. In Geographic Information Systems: The Microcomputer and Modern Cartography, edited by D. R. F. Taylor (Oxford: Pergamon Press), pp. 129 - 150.

Cartwright, B. A., and Collett, T. S., 1987, Landmark maps for honeybees. Biological Cybernetics, 57: 85–93.

Chang H.J., and Freeman W.J., 1996, Parameter optimization in models of the olfactory system, Neural Networks, Vol. 9, pp. 1-14.

Chase, W.G., 1983, Spatial representations of taxi drivers. In Rogers, R. and J. A. Sloboda, J.A., Eds., Acquisition of Symbolic Skills. New York.: Plenum, pp. 111-136.

Cheng, K., 1989, The vector sum model of pigeon landmark use. Journal of Experimental Psychology: Animal Behavioral Processes, 15: 366–375.

Chown, E., Kaplan, S., and Kortenkamp, D., 1995, Prototypes, location, and associative networks (PLAN): toward a unified theory of cognitive mapping. Cognitive Science, 19: 1–51.

Couclelis, H., 1986, Artificial intelligence in geography: Conjectures on the shape of things to come. The Professional Geographer, 38, 1- 11.

Cutting, J.E., 1996, Wayfinding from multiple source of local information in retinal flow. Journal of Experimental Psychology, 22(5), 1299-1313.

Darken, R. P., and Allard, T., 1999, Spatial Orientation and Wayfinding in Large-Scale Virtual Spaces: An Introduction, Presence Teleoperators and Virtual Environments, 7(2), pp. 101-107.

Darken, R. P., Allard, T., and Achille, L. B., 1998, Spatial Orientation and Wayfinding in Larger-Scale Virtual Spaces: An Introduction, Presence, 7(2), pp. 101-107.

Darken, R. P., Cockayne, W.R., and Carmein, D., 1997, The Omni-Directional Treadmill: A Locomotion Device for Virtual Worlds. Proc. of UIST'97, pp. 213-221.

De Luca A., Mattone, R., and Giordano, P.R., 2007, Acceleration-level control of the CyberCarpet. 2007 IEEE International Conference on Robotics and Automation, Roma, I, pp. 2330-2335.

Davis, E., 1983, The MERCATOR representation of spatial knowledge. Proceedings of Eighth International Conference on Artificial Intelligence, IJCAI-83, pp. 295 - 301.

Davis, E., 1986, Representing and Acquiring Geographic Knowledge. Research Notes in Artificial Intelligence (Los Altos, CA: Morgan Kaufman Publishers).

Downs, R.M., and Stea, D., 1973, Cognitive maps and spatial behaviour. Process and products. In R.M. Downs and D. Stea (Eds.), Image and environment: Cognitive mapping and spatial behaviour (pp.8-26). Chicago: Aldine.

Earnshaw, R. A., Gigante, M. A., and Jones, H., editors, 1993, Virtual Reality Systems. Academic Press, 1993.

Elliott, R. J., and Lesk, M. E., 1982, Route finding in street maps by computers and people. Proceedings of AAAI-82 Conference.

Engebretson, P. H., and Huttenlocher, J., 1996, Bias in spatial location due to categorization: Comment on Tversky and Schiano. Journal of Experimental Psychology: General, 125(1) 96-108.

Freeman, H., and Ahn, J., 1984, AUTONAP - An expert system for automatic name placement. In Proceedings, International Symposium on Spatial Data Handling, Zurich, pp. 544 - 569.

Freeman, W.J., 1975, Mass Action in the Nervous System. New York NY: Academic Press.

Freeman, W. J., 2000, How Brains Make up their Minds (Columbia: Columbia University Press).

Freeman, W.J., 2000, Neurodynamics. An exploration of mesoscopic brain dynamics. London, UK. Springer Verlag.

Freeman,W. J., Chang, H. J., Burke, B. C., Rose, P. A., and Badler, J., 1997, Taming chaos: stabilization of aperiodic attractors by noise. IEEE Transactions on Circuits and Systems, 44: 989–996. Gallistel, C. R., 1990, The organization of learning. Cambridge, MA: MIT Press.

Gopal, S., Klatzky, R. L., and Smith, T. R., 1989, NAVIGATOR- a psychologically based model of environmental learning through navigation. Journal of Environmental Psychology, 9, 309 - 331.

Goldin, S.E., and Thorndyke, P.W., 1982, Simulating Navigation for Spatial Knowledge Acquisition. Human Factors, 24(4), 457-471.

Golledge, R. G., Ruggles, A. J., Pellegrino, J.W., and Gale, N. D., 1993, Integrating route knowledge in an unfamiliar neighborhood: Along and across route experiments. Journal of Environmental Psychology, 13, 1 - 15.

Hasselmo, M. E., Hay, J., Ilyn, M., and Gorchetchnikov, A., 2002, Neuromodulation, theta rhythm and rat spatial navigation. Neural Networks, 15: 689–707.

Herrick, C.J., 1948, The Brain of the Tiger Salamander. Chicago IL: University of Chicago Press.

Hirose, M., and Yokoyama, K., 1997, Synthesis and transmission of realistic sensation using virtual reality technology. Transactions of the Society of Instrument and Control Engineers, vol.33, no.7, pp. 716-722.

Hollerbach, J.M., Xu, Y., Christensen, R., and Jacobsen, S.C., 2000, Design specifications for the second generation Sarcos Treadport locomotion interface. Haptics Symposium, Proc. ASME Dynamic Systems and Control Division, DSC-Vol. 69-2, Orlando, Nov. 5-10, 2000, pp. 1293-1298.

Iwata, H., and Fujji, T., 1996, Virtual Preambulator: A Novel Interface Device for Locomotion in Virtual Environment. Proc. of IEEE VRAIS'96, pp. 60-65.

Iwata, H., Yano, H., Fukushima, H., and Noma, H., 2005, CirculaFloor, IEEE Computer Graphics and Applications, Vol.25, No.1. pp. 64-67.

Iwata, H, Yano, H., and Tomioka, H., 2006, Powered Shoes, SIGGRAPH 2006 Conference DVD (2006).

Iwata, H, Yano, H., and Tomiyoshi, M., 2007, String walker. Paper presented at SIGGRAPH 2007.
Iwata, H., and Yoshida, Y., 1999, Path Reproduction Tests Using a Torus Treadmill. PRESENCE, 8(6), 587-597.

Kay, L.M., and Freeman, W.J., 1998, Bidirectional processing in the olfactory-limbic axis during olfactory behavior. Behav. Neurosci. 112: 541-553.

K. Lynch, 1960, The Image of the City. MIT Press, Cambridge, MA.

Kitchin, R., 1996, Methodological convergence in cognitive mapping research: investigating configurational knowledge. Journal of Environmental Psychology, 16, 163-185.

Kitchin, R., 1998, "Out of place", "knowing one's place": Towards a spatialised theory of disability and social exclusion. Disability and Society, 13, 343-356.

Kitchin, R. M., and Jacobson, R. D., 1997, Techniques to collect and analyze the cognitive map knowledge of persons with visual impairment or blindness: issues of validity. Journal of Visual Impairment and Blindness, 91, 393-400.

Klatzky, R. L., Golledge, R. G., Loomis, J. M., Cicinelli, J. G., and Pellegrino, J. W., 1995, Performance of blind and sighted persons on spatial tasks. Journal of Visual Impairment & Blindness, 89, 70-82.

Kohonen, T., 1977, Associative memory. Berlin: Springer.

Kozma, R., and Freeman, W.J., 2001, Chaotic resonance: Methods and applications for robust classification of noisy and variable patterns. Int. J. Bifurcation and Chaos, 11(6): 2307-2322.

Kozma, R., et al., 2001, Emergence of un-correlated common-mode oscillations in the sensory cortex. Neurocomputing, 38-40: 747-755.

Kozma, R., and Freeman, W. J., 2003, Basic principles of the KIV model and its application to the Q4 navigation problem. Journal of Integrative Neuroscience, 2: 1–21.

Kozma, R., Freeman W.J., and Erdi, P., 2003, The KIV model - nonlinear spatio-temporal dynamics of the primordial vertebrate forebrain. Neurocomputing. 52-4, pp. 819-826.

Kuipers, B., 1978, Modeling spatial knowledge. Cognitive Science, 2: 129–153.

Kuipers, B., 1979, Cognitive modeling of the map user. In Proceedings, First International Study Symposium on Topological Data Structures for Geographic Information Systems, Volume 7, edited by G. Dutton (Spatial Semantics: Understanding and Interacting with Map Data), pp. 1 - 11.

Kuipers, B., 1982, The `Map in the head' metaphor. Environment and Behavior, 14, 202 - 220.

Kuipers, B., 1983, The cognitive map: could it have been any other way? In Spatial Orientation: Theory, Research, and Application, edited by H. L. Pick, Jr and L. P. Acredolo (New York: Plenum Press), pp. 345 - 359.

Levenick, J. R., 1991, NAPS: a connectionist implementation of cognitive maps. Connection Science, 3(2): 107–126.

Mark, D. M., 1985, Finding simple routes: `Ease of description' as an objective function in automated route selection. In Proceedings, Second Symposium on Artificial Intelligence Applications (IEEE), Miami Beach, pp. 577 - 581.

Mark, D.M., and McGranaghan, M., 1986, Effective provision of navigation assistance for drivers: A cognitive science approach. Proceedings, Auto-Carto London, September 14 - 19, 2, 399 - 408.

Mark, D. M., Gould, M. D., and McGranaghan, M., 1987, Computerized navigation assistance for drivers. The Professional Geographer, 39, 215 - 220.

Mark, D. M., Svorou, S., and Zubin, D., 1987, Spatial terms and spatial concepts: Geographic, cognitive, and linguistic perspectives. In Proceedings, International Symposium on Geographic Information Systems: T he Research Agenda Crystal City, Virginia, 2, 101 – 112

Marr, D., 1982, Vision: A Computational Investigation into the Human Representation and Processing of Visual Information (New York: Freeman).

McGranaghan,M., Mark, D. M., and Gould, M. D., 1987, Automated provision of navigation assistance to drivers. The American Cartographer, 14, 121 - 138.

McNamara T. P., Hardy J. K., and Hirtle S. C., 1989, Subjective Hierarchies in Spatial Memory. J. of Experimental Psychology: Learning, Memory and Cognition, 15, pp. 211-227.

Munro, P., and Hirtle, S. C., 1989, An interactive activation model for priming of geographical information. In Proceedings 11th Annual Conference Cognitive Science Society, (Hillsdale: Erlbaum), pp. 773 - 780.

Noma, H., and Miyasato, T., 1998, Design for Locomotion Interface in a Large Scale Virtual Environment, ATLAS: ATR Locomotion Interface for Active Self Motion. 7th Annual Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. The Winter Annual Meeting of the ASME. Anaheim, USA.

Passini, R., 1984, Wayfinding in Architecture. New York: Van Nostrand Reinhold Company Inc.
Reid, A. K., and Staddon, J. E. R., 1998, A dynamic route finder for the cognitive map. Psychological Review, 105: 585–601.

Riesbeck, C. K., 1980, `You can't miss it': Judging the clarity of directions. Cognitive Science, 4, 285 - 303.

Samsonovich A, McNaughton BL, 1997, Path integration and cognitive mapping in a continuous attractor neural network model. J Neurosci 17:5900-5920.

Schiff, S.J., 1994, Controlling chaos in the brain. Nature, 370, 615-620.

Schmajuk, N. A., and Thieme, A. D., 1992, Purposive behavior and cognitive mapping: an adaptive neural network. Biological Cybernetics, 67: 165–174.

Seigel, A. W., and White, S. H., 1975, The development of spatial representations of large-scale environments. In H. W. Reese (Ed.), Advances in child development and behaviour (pp. 9–55). New York: Academic Press.

Sholl, M. J., 1987, Cognitive maps as orienting schemata. Journal of Experimental Psychology: Learning, Memory, and Cognition, 13(4), 615–628.

Sibert, L., Templeman, J., Page, R., Barron, J., McCune, J., and Denbrook, P., 2004, Initial Assessment of Human Performance Using the Gaiter Interaction Technique to Control Locomotion in Fully Immersive Virtual Environments. (Technical Report) Washington, D.C: Naval Research Laboratory.
Skarda, C.A., and Freeman, W.J., 1987, How brains make chaos in order to make sense of the world. Behavioral & Brain Sci., 10:161-195.

Stevens, A., and Coupe, P., 1978, Distortions in judged spatial relations. Cognitive Psychology, 10, 422-437.

Streeter, L. A., Vitello, D., and Wonsiewicz, S. A., 1985, How to tell people where to go: Comparing navigational aids. International Journal of Man/Machine Studies, 22, 549 - 562.

Streeter, L. A., and Vitello, D., 1986, A profile of drivers' map-reading abilities. Human Factors, 28, 223 - 239.

Talmy, L., 1983, How language structures space. In Spatial Orientation: Theory, Research and Application, edited by H. L. Pick, Jr. and L. P. Acredolo (New York: Plenum), pp. 225 – 282

Timpf, S., and Kuhn, W., 2003, Granularity Transformations in Wayfinding. In Freksa, C., Brauer, W., Habel, C., and Wender, K. (Eds.), Spatial Cognition III: Routes and Navigation, Human Memory and Learning, pp. 77-88.

Tolman, E. C., 1948, Cognitive maps in rats and men. In R. M. Downs & D. Stea (Eds.), Image and environment, cognitive mapping and spatial behavior (pp. 27–50).s Chicago: Edward Arnold.

Tolman, E. C., Ritchie, B. F., and Kalish, D., 1946, Studies in spatial learning. I. Orientation and the short-cut. J. exp. Psychol. 36, 13-24.

Trullier, O., and Meyer, J. A., 1997, Place sequence learning for navigation. In W. Gerstner, A.

Germond, M. Hasler and J.-D. Nicoud (eds) Artificial Neural SNetworks-ICANN'97, Lausanne, Q7 witzerland (Berlin: Springer), pp. 757–762.

Trullier, O., Wiener, S., Berthoz, A., and Meyer, J.-A., 1997, Biologically-based artificial navigation systems: review and prospects. Progress in Neurobiology, 51: 483–544.

Tversky, B., 1981, Distortions in memory for maps. Cognitive Psychology, 13, 407-433.

Viana Di Prisco, G., and Freeman, W.J., 1985, Odor-related bulbar EEG spatial pattern analysis during appetitive conditioning in rabbits. Behav. Neurosci. 99: 962-978.

Voicu, H., and Schmajuk, N. A., 2001, Three-dimensional cognitive mapping with a neural network. Robotics and Autonomous Systems, 35: 23–36.

Voicu, H., and Schmajuk, N. A., 2000, Exploration, navigation and cognitive mapping. Adaptive Behavior, 8(3/4), 207–223.

Voicu, H., and Schmajuk, N. A., 2002, Latent learning, shortcuts and detours: a computational model. Behavioural Processes, 59: 67–86.

Wilkie, D. M., and Palfrey, R., 1987, A computer simulation model of rats' place navigation in the Morris water maze. Behavior Research Methods, Instruments and Computers, 19: 400–403.

Wender, K. F., 1989, Connecting analog and verbal representations for spatial relations. Paper presented at the 30th Annual Meeting of the Psychonomic Society, Atlanta, Georgia.

Weston, L. and S. Handy, 2004, Mental Maps, in Handbook of Transport Geography and Spatial Systems, D.A. Hensher, et al., Editors. Elevesier, Amsterdam, 2004.

Yeap, W. K., 1988, Towards a computational theory of cognitive maps. Artificial Intelligence, 34, 297 - 360.

Zhang X. A multiscale progressive model on virtual navigation, 2008, International Journal of Human Computer Studies, 66 (4), pp. 243-256.

Zavoli, W. B., Honey, S. K., and White, M. S., 1985, A land vehicle navigation system supported by digital map data base. ADPA Artificial Intelligence and Robotics Symposium.

TMRF e-Book
Advanced Knowledge Based Systems:
Model, Applications & Research
(Eds. Sajja & Akerkar), Vol. 1, pp 126 – 159, 2010

Chapter 7

Bio-Inspired Algorithms for Fuzzy Rule-Based Systems

Bahareh Atoufi * and Hamed Shah-Hosseini †

Abstract In this chapter we focus on three bio-inspired algorithms and their combinations with fuzzy rule based systems. Rule Based systems are widely being used in decision making, control systems and forecasting. In the real world, much of the knowledge is imprecise and ambiguous but fuzzy logic provides for systems better presentations of the knowledge, which is often expressed in terms of the natural language. Fuzzy rule based systems constitute an extension of classical Rule-Based Systems, because they deal with fuzzy rules instead of classical logic rules. Since bio-inspired optimization algorithms have been shown effective in improving the tuning the parameters and learning the Fuzzy Rule Base, our major objective is to represent the application of these algorithms in improving the Knowledge Based Systems with focus on Rule-Base Learning.
We first introduce the Evolutionary Computation and Swarm Intelligence topics as a subfield of Computational Intelligence dealing with Combinatorial Optimization problems. Then, three following bio-inspired algorithms, i.e. Genetic Algorithms, Ant Colony Optimization, and Particle Swarm Optimization are explained and their application in improving the knowledge Based Systems are presented.

INTRODUCTION

The generation of the Knowledge Base (KB) of a Fuzzy Rule-Based System (FRBS) [Mamdani E.H., 1982] presents several difficulties because the KB depends on the concrete application, and this makes the accuracy of the FRBS directly depend on its composition. Many approaches have been proposed to automatically learn the KB from numerical information. Most of them have focused on the Rule Base (RB) learning, using a predefined Data Base (DB) [Cordón et al. 1998]. This operation mode makes the DB have a significant influence on the FRBS performance. In fact, some studies have shown that the system performance is much more sensitive to the choice of the semantics in the DB than to the composition of the RB. The usual solution for improving the FRBS performance in dealing with the DB components involves a tuning process of the preliminary DB definition once the RB has been derived. This process only adjusts the membership function definitions and does not modify the

* Department of Electrical and Computer Engineering, Shahid Beheshti University, G.C., atoufi@gmail.com
† Corresponding author: Department of Electrical and Computer Engineering, Shahid Beheshti University, G.C., tasom2002@yahoo.com.com, h_shahhosseini@sbu.ac.ir

number of linguistic terms in each fuzzy partition since the RB remains unchanged. In contrast to this, *a posteriori* DB learning, there are some approaches that learn the different DB components *a priori*. The objective of this chapter is to introduce methods to automatically generate the KB of an FRBS based on learning approach. We show the application of Evolutionary Computation (EC) [Back T., Schwefel H.P. 1986] and Swarm Intelligence (Genetic Algorithms (GA) [Goldberg D.E., 1989], Ant Colony Optimization (ACO) [Dorigo M., 1992], and Particle Swarm Optimization (PSO)[Kennedy J., Eberhart R.C. 1995]) in improving the Rule-base learning.

The outline of the chapter is as follows: in Section 1, we introduce the Evolutionary Computing concept. In addition to a brief history, we give an introduction to some of the biological processes that have served as an inspiration and that have provided rich sources of ideas and metaphors to researchers. The GA as part of EC is described in a subsection of this part. In Section 2, Swarm Intelligence is described with focus on introducing ACO and PSO as the most popular optimization algorithms based on Swarm Intelligence used for fuzzy rule bases. Section 3 includes an introduction to Fuzzy Rule-Based Systems as an extension of classical Rule-Based systems. Genetic Fuzzy Rule-Based Systems are introduced in Section 4. Generating the Knowledge Base of a Fuzzy Rule-Based System by the Genetic Learning of the Data Base is discussed here. In this part, we give the reader an idea about Genetic Tuning and Genetic Learning of Rule Bases. Section 5 is devoted to introduce all the aspects related to ACO algorithms specialized to the FRL (Fuzzy Rule Learning) problem. Here, the learning task is formulated as an optimization problem and is solved by the ACO. Finally, in Section 6, application of PSO for Fuzzy Rule-Based System is discussed.

1. EVOLUTIONARY COMPUTATION

Nearly three decades of research and applications have clearly demonstrated that the mimicked search process of natural evolution can yield very robust, direct computer algorithms, although these imitations are crude simplifications of biological reality. The resulting Evolutionary Algorithms are based on the collective learning process within a population of individuals, each of which represents a search point in the space of potential solutions to a given problem. The population is arbitrarily initialized, and it evolves towards better and better regions of the search space by means of randomized processes of selection (which is deterministic in some algorithms), mutation, and recombination (which is completely omitted in some algorithmic realizations). The environment (given aim of the search) delivers some information about quality (fitness value) of the search points, and the selection process favors those individuals of higher fitness to reproduce more often than worse individuals. The recombination mechanism allows the mixing of parental information while passing it to their descendants, and mutation introduces innovation into the population.

$f : R^n \to R$ denotes the objective function to be optimized, and without loss of generality we assume a minimization task in the following. In general, fitness and objective function values of an individual are not required to be identical, such that fitness $\Phi : I \to R$ (I being the space of individuals) and f are distinguished mappings (but f is always a component of Φ). While $\vec{a} \in I$ is used to denote an individual, $\vec{x} \in R^n$ indicates an object variable vector. Furthermore $\mu \leq 1$ denotes the size of the parent population and $\lambda \geq 1$ is the offspring population size, i.e. the number of individuals created by means of recombination and mutation at each generation (if the life time of individuals is limited to one generation, i.e. selection is not elitist, $\lambda \geq \mu$ is a more reasonable assumption). A population at generation t, $P(t) = \{\vec{a}_1(t),...,\vec{a}_\mu(t)\}$ consists of individuals, $\vec{a}_i(t) \in I$, and $r_{\Theta_r} : I^\mu \to I^\lambda$ denotes the recombination operator which might be controlled by

additional parameters summarized in the set Θ_r .Similarly, the mutation operator $m_{\Theta_m} : I^{\lambda} \to I^{\lambda}$ modifies the offspring population, again being controlled by some parameters Θ_m . Both mutation and recombination, though introduced here as macro-operators transforming populations into populations, can essentially be reduced to local operators $m'_{\Theta_m} : I \to I$ and $r'_{\Theta_r} : I^{\mu} \to I$ that create one individual when applied. These local variants will be used when the particular instances of the operators are explained in subsequent Sections. Selection $s_{\Theta_s} : (I^{\lambda} \cup I^{\mu+\lambda}) \to I^{\mu}$ is applied to choose the parent population of the next generation. During the evaluation steps the fitness function $\Phi : I \to R$ is calculated for all individuals of a population, and $\iota : I^{\mu} \to \{true, false\}$ is used to denote the termination criterion. The resulting algorithmic description is given below.

$t := 0;$
initialize $P(0) := \{\vec{a}_1(0), \ldots, \vec{a}_\mu(0)\} \in I^{\mu};$
evaluate $P(0) : \{\Phi(\vec{a}_1(0)), \ldots, \Phi(\vec{a}_\mu(0))\};$
while $(\iota(P(t)) \neq$ **true**$)$ **do**
recombine: $P'(t) := r_{\Theta_r}(P(t));$
mutate: $P''(t) := m_{\Theta_m}(P'(t));$
evaluate $P''(t) : \{\Phi(\vec{a}''_1(t)), \ldots, \Phi(\vec{a}''_\lambda(t))\};$
select: $P(t+1) := s_{\Theta_s}(P''(t) \cup Q);$
$t := t + 1;$
od

Here, $Q \in \{0, P(t)\}$ is a set of individuals that are additionally taken into account during the selection step. The evaluation process yields a multi-set of fitness values, which in general are not necessarily identical to objective function values. However, since the selection criterion operates on fitness instead of objective function values,
fitness values are used here as result of the evaluation process. During the calculation of fitness, the evaluation of objective function values is always necessary, such that the information is available and can easily be stored in an appropriate data structure. Three main streams of instances of this general algorithm, developed independently of each other, can nowadays be identified: Evolutionary Programming (EP), originally developed by L. J. Fogel, A. J. Owens, and M. J.Walsh in the U.S., Evolution Strategies (ESs), developed in Germany by I. Rechenberg, and Genetic Algorithms (GAs) by J. Holland in the U.S. as well, with refinements by D. Goldberg. Each of these main stream algorithms have clearly demonstrated their capability to yield good approximate solutions even in case of complicated multimodal, discontinuous, non-differentiable, and even noisy or moving response surfaces of optimization problems. Within each research community, parameter optimization has been a common and highly successful theme of applications. Evolutionary Programming and especially GAs, however, were designed with a very much broader range of possible applications in mind and confirmed their wide applicability by a variety of important examples in fields like machine learning, control, automatic programming, planning, and etc. Here, we focus on GA as a successful optimization method which has been shown effective in improving the tuning the parameters and learning the Fuzzy Rule Base.

1.1 Genetic Algorithm

GA is a part of evolutionary computing. In fact, they are adaptive heuristic search algorithms premised on the evolutionary ideas of natural selection and genetic. The basic concept of GA is designed to simulate processes in the natural system necessary for evolution, specifically as you can guess, GAs often follow the principles first laid down by Charles Darwin of survival of the fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem.
First pioneered by John Holland in the 60s, GA has been widely studied, experimented and applied in many fields in engineering worlds. Not only does GA provide an alternative method to problem solving, it consistently outperforms other traditional methods in most of the problems. Many of the real world problems involved finding optimal parameters, which might prove difficult for traditional methods but ideal for GA. However, because of its outstanding performance in optimization, GA has been regarded as a function optimizer. In fact, there are many ways to view GA. Perhaps, most users come to GA, looking for a problem solver, but this is a restrictive view. In fact, GA can be examined as a number of different things:

- GA as problem solvers
- GA as challenging technical puzzle
- GA as basis for competent machine learning
- GA as computational model of innovation and creativity
- GA as computational model of other innovating systems
- GA as guiding philosophy

However, due to various constraints, we would only be looking at GAs as problem solvers here.
The problem addressed by GA is to search a space of candidate hypotheses to identify the best hypothesis. In GA, the "best hypothesis" is defined as the one that optimizes a predefined numerical measure for the problem at hand, called the hypothesis *fitness*. Although different implementations of GA vary in their details, they typically share the following structure: The algorithm operates by iteratively updating a pool of hypotheses, called the population. In each iteration, all members of the population are evaluated according to the fitness function. A new population is then generated by probabilistically selecting the fittest individuals from the current population. Some of these selected individuals are carried forward into the next generation population intact. Others are used as the basis for creating new offspring individuals by applying genetic operations such as *crossover* and *mutation*.
A prototypical GA is described in Table 1.1 The input to this algorithm includes the fitness function for ranking candidate hypotheses, a threshold defining an acceptable level of fitness for terminating the algorithm, the size of the population to be maintained, and parameters that determine how successor populations are to be generated: the fraction of the population to be replaced at each generation and the mutation rate. Notice in this algorithm each iteration through the main loop produces a new generation of hypotheses based on the current population.

Table 1.1. A prototypical GA. A population containing ***p*** hypotheses is maintained. In each iteration, the successor population ***P***s is formed by probabilistically selecting current hypotheses according to their fitness and by adding new hypotheses. New hypotheses are created by applying a crossover operator to pairs of fittest hypotheses and by creating single point mutations in the resulting generation of hypotheses. This process is iterated until sufficiently fit hypotheses are discovered. Typical crossover and mutation operators are defined in a subsequent Table.

GA (Fitness, Fitness-threshold, p, r, m)
Fitness: A function that assigns an evaluation score, given a hypothesis.
Fitness-threshold: A threshold specifying the termination criterion.
p: The number of hypotheses to be included in the population.
r: The fraction of the population to be replaced by Crossover at each step.
m: The mutation rate.

Initialize population: P ← Generate ***p*** hypotheses at random
Evaluate: For each hypothesis h_i in **P,** compute **Fitness(**h_i**)**
While ($\max_i Fitness(h_i)$ < **Fitness-threshold)** do

Create a new generation, denoted by *Ps*:

Select: Probabilistically select (1 - *r)p* members of ***P*** to be included in ***Ps***. The probability Pr(h_i) of selecting hypothesis h_i from ***P*** is given by

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{P} Fitness(h_i)}$$

Crossover: Probabilistically select $r \cdot p$ pairs of hypotheses from ***Ps***, according to Pr(h_i) given above. For each pair $[h_1, h_2]$, produce two offspring by applying the Crossover operator. Add all offspring to ***Ps***
Mutate: Choose m percent of the members of ***Ps***, with uniform probability. For each, flip one randomly-selected bit in its representation.
Update: ***P*** ← ***Ps***
Evaluate***:*** for each hypothesis h_i in ***P,*** compute **Fitness(**h_i**)**

Return the hypothesis from ***P*** that has the highest fitness.

First, a certain number of hypotheses from the current population are selected for inclusion in the next generation. These are selected probabilistically, where the probability of selecting hypothesis h_i is given by

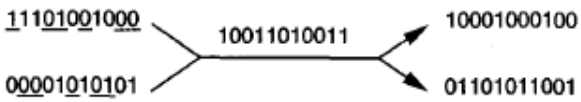
$$\Pr(h_i) = \frac{Fitness\,(h_i)}{\sum_{j=1}^{P} Fitness\,(h_i)} \quad (1)$$

Thus, the probability that a hypothesis will be selected is proportional to its own fitness and is inversely proportional to the fitness of the other competing hypotheses in the current population.
Once these members of the current generation have been selected for inclusion in the next generation population, additional members are generated using a crossover operation. Crossover, defined in detail in [Mitchel, 1997], takes two parent hypotheses from the current generation and creates two offspring hypotheses by recombining portions of both parents. The parent hypotheses are chosen probabilistically from the current population. Again, using the probability function given by Eq. 1, after new members have been created by this crossover operation, the new generation population now contains the desired number of members.
At this point, a certain fraction *m* of these members are chosen at random and random mutations are all performed to alter these members. This GA thus performs a randomized, parallel beam search for hypotheses that perform well according to the fitness function.
The generation of successors in a GA is determined by a set of operators that recombine and mutate selected members of the current population. Typical GA operators for manipulating bit string hypotheses are illustrated in Table 1.1. These operators correspond to idealized versions of the genetic operations found in biological evolution. The two most common operators are *crossover* and *mutation*. The *crossover operator* produces two new offspring from two parent strings, by copying selected bits from each parent. A second type of operator produces offspring from a single parent. In particular, the *mutation operator* produces small random changes to the bit string by choosing a single bit at random, then changing its value. Mutation is often performed after crossover has been applied as depicted in our prototypical algorithm summarized in Table 1.1.
Note that some GA systems employ additional operators, especially operators that are specialized to the particular hypothesis representation used by the system. Table 1.2 illustrates examples of some GA common operators.

Table 1.2. Common operators for GA. These operators form offspring of hypotheses represented by bit strings. The crossover operators create two descendants from two parents, using the crossover mask to determine which parent contributes to which bits. Mutation creates a single descendant from a single parent by changing the value of a randomly chosen bit.

	Initial strings	Crossover Mask	Offspring
Single-point crossover:	11101001000 00001010101	11111000000	11101010101 00001001000
Two-point crossover:	11101001000 00001010101	00111110000	11001011000 00101000101
Uniform crossover:	11101001000 00001010101	10011010101	11001000100 00101011001
Mutation:	11101001000	00010000000	11111001000

Uniform crossover:



Point mutation:

The fitness function defines the criterion for ranking potential hypotheses and for probabilistically selecting them for inclusion in the next generation population. If the task is to learn classification rules, then the fitness function typically has a component that scores the classification accuracy of the rule over a set of provided training examples. Often other criteria may be included as well, such as the complexity or generality of the rule. More generally, when the bit-string hypothesis is interpreted as a complex procedure (e.g., when the bit string represents a collection of if-then rules that will be chained together to control a robotic device), the fitness function may measure the overall performance of the resulting procedure rather than performance of individual rules.

2. SWARM INTELLIGENCE

Swarm Intelligence [Kennedy and Eberhart, 2001] provides a useful paradigm for implementing adaptive systems. In this sense, it is an extension of EC. Swarm Intelligent is the emergent collective intelligence of groups of simple agents. At a time when the world is becoming so complex that no single agent can understand it, Swarm Intelligence offers an alternative way of designing intelligent systems [Bonabeau et al. 1999]. Two of the most popular algorithms based on Swarm Intelligence are ACO and PSO, which we are going to discuss their application for Knowledge-Based Systems.

2.1 Ant Colony optimization

ACO is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs.
This algorithm is a member of ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimizations. Initially proposed by Marco Dorigo in 1992 in his PhD thesis, the first algorithm was aiming to search for an optimal path in a graph; based on the behavior of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of Numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behavior of ants.
The characteristic of ACO algorithms is their explicit use of elements of previous solutions. In fact, they drive a constructive low-level solution, but including it in a population framework and randomizing the construction in a Monte Carlo way. A Monte Carlo combination of different solution elements is suggested also by GA, but in the case of ACO the probability distribution is explicitly defined by previously obtained solution components.
Here, we describe the common elements of the heuristics following the ACO paradigm and outline some of the variants proposed. Later, we present the application of ACO algorithms to learning Fuzzy Rules.
ACO is a class of algorithms, whose first member, called Ant System (AS), was initially proposed by Colorni, Dorigo and Maniezzo.
The functioning of an ACO algorithm can be summarized as follows. A set of computational concurrent and asynchronous agents (a colony of ants) moves through states of the problem

corresponding to partial solutions of the problem to solve. They move by applying a stochastic local decision policy based on two parameters, called *trails* and *attractiveness*. By moving, each ant incrementally constructs a solution to the problem. When an ant completes a solution, or during the construction phase, the ant evaluates the solution and modifies the pheromone trail value on the components used in its solution. This pheromone information will direct the search of the future ants. Furthermore, an ACO algorithm includes two more mechanisms: *trail evaporation* and, optionally, *daemon actions*. Trail evaporation decreases all trail values over time, in order to avoid unlimited accumulation of trails over some component. Daemon actions can be used to implement centralized actions which cannot be performed by single ants, such as the invocation of a local optimization procedure, or the update of global information to be used to decide whether to bias the search process from a non-local perspective.
More specifically, an *ant* is a simple computational agent, which iteratively constructs a solution for the instance to solve. Partial problem solutions are seen as *states*. At the core of the ACO algorithm lies a loop, where at each iteration, each ant *moves* (performs a *step*) from a state ι to another one ψ, corresponding to a more complete partial solution. That is, at each step, each ant *k* computes a set $A_k^{\sigma}(\iota)$ of feasible expansions to its current state, and moves to one of these in probability. The probability distribution is specified as follows. For ant *k*, the probability $p_{\iota\psi}^k$ of moving from state ι to state ψ depends on the combination of two values:

the *attractiveness* $\eta_{\iota\psi}$ of the move, as computed by some heuristic indicating the *a priori* desirability of that move[‡];

the pheromone *trail level* $\tau_{\iota\psi}$ of the move, indicating how efficient it has been in the past to make that particular move: it represents therefore an *a posteriori* indication of the desirability of that move.

Trails are *updated* usually when all ants have completed their solution, increasing or decreasing the level of trails corresponding to moves that were part of "good" or "bad" solutions, respectively.

The importance of the original Ant System (AS) resides mainly in being the prototype of a number of ant algorithms, which collectively implement the ACO paradigm. The components of AS are specifies as follows. The move probability distribution defines probabilities $p_{\iota\psi}^k$ to be equal to 0 for all moves which are infeasible (i.e., they are in the *tabu* list of ant *k*, that is a list containing all moves which are infeasible for ants *k* starting from state ι, otherwise they are computed by means of Eq. 2, where α and β are user defined parameters (another parameter, $0 \le \rho \le 1$):

[‡] The attractiveness of a move can be effectively estimated by means of lower bounds (upper bounds in the case of maximization problems) on the cost of the completion of a partial solution. In fact, if a state ι corresponds to a partial problem solution it is possible to compute a lower bound on the cost of a complete solution containing ι. Therefore, for each feasible move ι,ψ, it is possible to compute the lower bound on the cost of a complete solution containing ψ: the lower the bound the better the move.

$$p_{\iota\psi}^{k} = \begin{cases} \dfrac{\tau_{\iota\psi}^{\alpha} \times \eta_{\iota\psi}^{\beta}}{\sum_{\zeta \notin tabu_k} \left(\tau_{\iota\zeta}^{\alpha} \times \eta_{\iota\zeta}^{\beta}\right)} & if\,(\iota\psi) \notin tabu_k \\ 0 \end{cases} \tag{2}$$

In En. 2, $tabu_k$ is the *tabu* list of ant *k*, while parameters α and β specify the impact of trail and attractiveness, respectively. After each iteration *t* of the algorithm, *i.e.,* when all ants have completed a solution, trails are updated by means of Eq. 3:

$$\tau_{\iota\psi}(\tau) = \rho\tau_{\iota\psi}(\tau - 1) + \Delta\tau_{\iota\psi} \tag{3}$$

where $\Delta\tau_{\iota\psi}$ represents the sum of the contributions of all ants that used move ($\iota\psi$) to construct their solution, ρ is a user-defined parameter called evaporation coefficient, and $\Delta\tau_{\iota\psi}$ represents the sum of the contributions of all ants that used move ($\iota\psi$) to construct their solution. The ants' contributions are proportional to the quality of the solutions achieved, i.e., the better solution is, the higher will be the trail contributions added to the moves it used.

For example, in the case of the TSP (Traveling Salesman's Problem), moves correspond to arcs of the graph, thus state ι could correspond to a path ending in node ι, the state ψ to the same path but with the arc (ιψ) added at the end and the move would be the traversal of arc (ιψ). The quality of the solution of ant *k* would be the length L_k of the tour found by the ant and Eq. 3 would become

$\tau_{ij}(t) = \tau_{ij}(t-1) + \Delta\tau_{ij}$, with

$$\Delta\tau_{\iota\psi} = \sum_{k-1}^{m} \Delta\tau_{\iota\psi}^{k} \tag{4}$$

Where *m* is the number of ants and $\Delta\tau_{ij}^{k}$ is the amount of trail laid on edge (ιψ) by ant *k*, which can be computed as

$$\tau_{\iota\psi}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{if ant k uses arc(ιψ) in its tour} \\ 0 \end{cases} \tag{5}$$

Q being a constant parameter.

The ant system simply iterates a main loop where *m* ants construct in parallel their solutions, thereafter updating the trail levels. The performance of the algorithm depends on the correct tuning of several parameters, namely: α and β, relative importance of trail and attractiveness, ρ, trail evaporation, $\tau_{ij}(0)$, initial trail level, *m*, number of ants, and *Q*, used for defining to be of high quality solutions with low cost. The algorithm is shown in Table 1.3.

Table1.3. Ant System (AS) algorithm

1. {Initialization}
   Initialize $\tau_{\iota\psi}$ and $\eta_{\iota\psi}$, $\forall (\iota\psi)$.

2. {Construction}
   For each ant k (currently in state ι) do
       repeat
           choose in probability the state to move into.
           append the chosen move to the k-th ant's set $tabu_k$.

       until ant k has completed its solution.
   end for

3. {Trail update}
   For each ant move $(\iota\psi)$ do
       compute $\Delta\tau_{\iota\psi}$
       update the trail matrix.
   end for

4. {Terminating condition}
   If not(end test) go to step 2

2.2. Particle Swarm Optimization

PSO is a population-based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. The initial intent of the particle Swarm concept was to graphically simulate the graceful and unpredictable choreography of a bird flock with the aim of discovering patterns that governs the ability of birds to fly synchronously, and to suddenly change direction with a regrouping in an optimal formation. From this initial objective, the concept evolved into a simple and efficient optimization algorithm [Andries and Engelbrecht, 2007; Kennedy and Eberhart, 2001].

The PSO belongs to the class of direct search methods used to find an optimal solution to an objective function (aka fitness function) in a search space. The PSO algorithm is simple, in the sense that the even the basic form of the algorithm yields results, it can be implemented by a programmer in a short duration, and it can be used by anyone with an understanding of objective functions and the problem at hand without needing an extensive background in mathematical optimization theory.

The particle swarm simulates a kind of social optimization. A problem is given, and also some way is given to evaluate a proposed solution, which is often in the form of a fitness function. A communication structure or social network is also defined, assigning neighbors for each individual to interact with. Then a population of individuals defined as random guesses at the problem solutions is initialized. These individuals are candidate solutions. They are also known as the particles, hence the name particle swarm. An iterative process to improve these candidate solutions is set in motion. The particles iteratively evaluate the fitness of the candidate solutions and remember the location where they had their best success. The individual's best solution is called the particle best or the local best. Each particle makes this information available to their neighbors. They are also able to see where their neighbors have had success. Movements through the search space are guided by these successes, with the population usually converging (by the end of a trial) to a solution, which is better than that of non-swarm approach using the same methods.

The swarm is typically modeled by particles in multidimensional space that have a position and a velocity. These particles fly through hyperspace (i.e. Rn) and have two essential reasoning capabilities: their memory of their own best position and knowledge of the global or their neighborhood's best. In a minimization optimization problem, problems are formulated so that "best" simply means the position with the smallest objective value. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. So a particle has the following information to make a suitable change in its position and velocity:

- A global best that is known to all and immediately updated when a new best position is found by any particle in the swarm.
- Neighborhood best that the particle obtains by communicating with a subset of the swarm.
- The local best, which is the best solution that the particle has seen.

The particle position and velocity update equations in the simplest form that govern the PSO are given by [Bonabeau et al. 1999]:

$$v_{ij} \leftarrow c_0 v_i + c_1 r_1 (globalbest_j - x_{ij}) + c_2 r_2 (localbest_{ij} - x_{ij}) + c_3 r_3 (neighborhoodbest_j - x_{ij})$$

$$x_{ij} \leftarrow x_{ij} + v_{ij}$$

As the swarm iterates, the fitness of the global best solution improves (decreases for minimization problem). It could happen that all particles being influenced by the global best eventually approach the global best, and from there on the fitness never improves despite many runs the PSO is iterated thereafter. The particles also move about in the search space in close proximity to the global best and not exploring the rest of search space. This phenomenon is called convergence. If the inertial coefficient of the velocity is small, all particles could slow down until they approach zero velocity at the global best. The selection of coefficients in the velocity-updating equations affects the convergence and the ability of the swarm to find the optimum. One way to come out of the situation is to reinitialize the particles positions at intervals when convergence is detected.

The algorithm presented below uses the global best and local bests but no neighborhood bests. Neighborhood bests allow parallel exploration of the search space and reduce the susceptibility of PSO to falling into local minima, but slow down convergence speed. Note that neighborhoods merely slow down the proliferation of new bests, rather than creating isolated sub swarms because of the overlapping of neighborhoods: to make neighborhoods of size three, say, particle 1 would only communicate with particles 2 through 5, particle 2 with 3 through 6, and so on. But, then a new best position discovered by particle 2's neighborhood would be communicated to particle 1's neighborhood at the next iteration of the PSO algorithm presented below. Smaller neighborhoods lead to slower convergence, while larger neighborhoods lead to faster convergence, with a global best representing a neighborhood consisting of the entire swarm.

A single particle by itself is unable to accomplish anything. The power is in interactive collaboration. Considering the algorithm shown in Table 1.4, one should note that ω is an inertial constant and its good values are usually slightly less than 1. c_1 and c_2 are constants that say how much the particle is directed towards good positions.

Table1.4. A basic canonical PSO algorithm

Let $f: R^n \rightarrow R$ be the fitness function that takes a particle's solution with several components in higher dimensional space and maps it to a single dimension metric. Let there be *n* particles, each

with associated positions $x_i \in R^m$ and velocities $v_i \in R^m$, i=1,…,n . Let $\hat{x}_i$ be the current best position of each particle i and let $\hat{g}$ be the global best.

Initialize x_i and v_i for all i. One common choice is to take $x_{ij} \in U[a_i, b_j]$ and $v_i = 0$ for all i and j=1,…,m where a_i, b_j are the limits of the search domain in each dimension, and U represents the Uniform distribution (continuous).

$\hat{x}_i \leftarrow x_i$ and $\hat{g} \leftarrow \arg\min_{xi} f(x_i)$, i=1,…,n.
While not converged:
For each particle $1 \le i \le n$:
Create random vectors $r_1, r_2 : r_{1j}$ and r_{2j} for all j, by taking r_{1j} , $r_{2j} \in$ U[0,1] for j=1,…,m.
Update the particle velocities:
$v_i \leftarrow \omega v_i + c_1 r_1 o(\hat{x}_i - x_i) + c_2 r_2 o(\hat{g} - x_i).$
Update the particle positions: $x_i \leftarrow x_i + v_i$.
Update the local bests if $f(x_i) < f(\hat{x}_i), \hat{x}_i \leftarrow x_i$.
Update the global best if $f(x_i) < f(\hat{g}), \hat{g} \leftarrow x_i$.
$\hat{g}$ is the optimal solution with fitness f($\hat{g}$).

They represent a "cognitive" and a "social" component, respectively, in that they affect how much the particle's personal (local) best and the global best (respectively) influence its movement. We usually take c_1, $c_2 \approx 2$. Also, r_1, r_2 are two random vectors with each component generally a uniform random number between zero and one. The operator $\circ$ indicates element-by-element multiplication i.e. the Hadamard matrix multiplication operator. There is a misconception arising from the tendency to write the velocity formula in a "vector notation" (see for example D.N. Wilke's papers). The original intent (see M.C.'s "Particle Swarm Optimization, 2006") was to multiply a new random component per dimension, rather than multiplying the same component with each dimension per particle. Moreover, r_1 and r_2 are supposed to consist of a single number, defined as $c_{\max}$, which normally has a relationship with ω (defined as c_1 in the literature) through a transcendental function, given the value $\varphi : c_1 = 1.0/(\varphi - 1.0 + (v_\varphi * v_\varphi) - (2.0 * v_\varphi))$; and $c_{\max} = c_1 * \varphi$. Optimal "confidence coefficients" are therefore approximately in the ratio scale of c_1 = 0.7 and $c_{\max}$ = 1.43.

3. FUZZY RULE-BASED SYSTEMS

3.1 Rule-Based Systems

Rule-Based Systems (RBSs) constitute the best currently available means for codifying the problem-solving know-how of human experts. Experts tend to express most of their problem-solving techniques in terms of a set of situation-action rules, and this suggests that RBSs should be the method of choice for building knowledge-intensive expert systems. Although many different techniques have emerged for organizing collections of rules into automated experts, all RBSs share certain key properties:

- They incorporate practical human knowledge in conditional if-then rules,
- Their skill increases at a rate proportional to the enlargement of their knowledge bases,

- They can solve a wide range of possibly complex problems by selecting relevant rules and then combining the results inappropriate ways,
- They adaptively determine the best sequence of rules to execute, and
- They explain their conclusions by retracing their actual lines of reasoning and translating the logic of each rule employed into natural language.

RBSs address a need for capturing, representing, storing, distributing, reasoning about, and applying human knowledge electronically. They provide a practical means of building automated experts in application areas where job excellence requires consistent reasoning and practical experience. The hallmark of these systems is the way they represent knowledge about plausible inferences and preferred problem-solving tactics. Typically, both sorts of know-how are represented as conditional rules.

Roughly speaking, an RBS consists of a knowledge base and an inference engine (Figure 1). The knowledge base contains rules and facts. Rules always express a conditional, with an antecedent and a consequent component. The interpretation of a rule is that if the antecedent can be satisfied the consequent can too. When the consequent defines an action, the effect of satisfying the antecedent is to schedule the action for execution. When the consequent defines a conclusion, the effect is to infer the conclusion. Because the behavior of all RBSs derives from this simple regimen, their rules will always specify the actual behavior of the system when particular problem-solving data are entered.

In doing so, rules perform a variety of distinctive functions:

- They define a parallel decomposition of state transition behavior, thereby inducing a parallel decomposition of the overall system state that simplifies auditing and explanation. Every result can thus be traced to its antecedent data and intermediate rule-based inferences.
- They can simulate deduction and reasoning by expressing logical relationships (conditionals) and definitional equivalences.
- They can simulate subjective perception by relating signal data to higher level pattern classes.
- They can simulate subjective decision making by using conditional rules to express heuristics.

As Figure 1 illustrates, a simple RBS consists of storage and processing elements, which are often referred to respectively as the knowledge-base and the inference engine.

```
If the plaintiff did receive an eye injury
    and there was just one eye that was injured
    and the treatment for the eye did require surgery
    and the recovery from the injury was almost complete
    and visual acuity was slightly reduced by the injury
    and the condition is fixed,
increase the injury trauma factor by $10,000.

If the plaintiff's injury did cause
        (a temporary disability of an important function)
    and the plaintiff's doctors were not certain about
        the disability being temporary
    and the plaintiff's recovery was almost complete
    and the condition is fixed,
increase the fear factor by $1,000 per day.

If the plaintiff did not wear glasses before the injury
    and the plaintiff's injury does require
        (the plaintiff to wear glasses),
increase the faculty loss factor by $1,500
    and increase the inconvenience factor by $1,500.
```

Figure 2. The Representation of Legal Heuristics

The basic cycle of an RBS consists of a select phase and an execute phase. During the execute phase, the system interprets the selected rule to draw inferences that alter the system's dynamic memory. System storage includes components for long-term static data and short-term dynamic data. The long term store, which is the knowledge-base, contains rules and facts. Rules specify actions the system should initiate when certain triggering conditions occur. These conditions define important patterns of data that can arise in working memory. The system represents data in terms of relations, propositions, or equivalent logical expressions. Facts define static and true propositions.

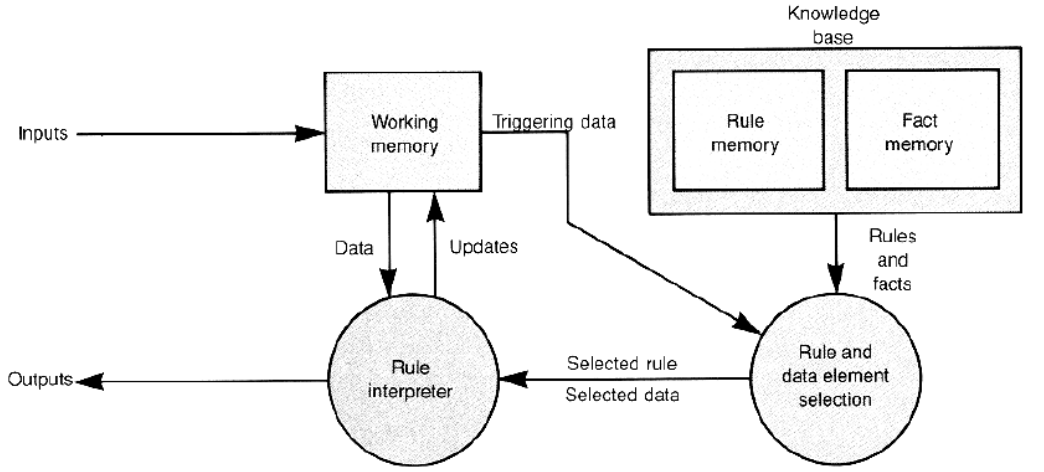


Figure 1.The Basic Features of a RBS

In contrast to conventional data-processing systems, most RBSs distribute their logic over numerous independent condition-action rules, monitor dynamic results for triggering patterns of data, determine their sequential behavior by selecting their next activity from a set of candidate triggered rules, and store their intermediate results exclusively in a global working memory.

3.2 Fuzzy Rule-Based Systems

Fuzzy logic was initiated in 1965 by Lotfi A. Zadeh, professor in Computer Science at the University of California in Berkeley. Since then, Fuzzy logic has emerged as a powerful technique for the controlling industrial processes, household and entertainment electronics, diagnosis systems and other expert systems. Rapid growth of this technology has actually started from Japan and then spread to the USA and Europe.

The motivation for fuzzy logic was expressed by Zadeh (1984) in the following way: "The ability of the human mind to reason in fuzzy terms is actually of a great advantage. Even though a tremendous amount of information is presented to the human senses in a given situation – an amount that would choke a typical computer – somehow the human mind has the ability to discard most of this information and to concentrate only on the information that is task relevant. This ability of the human mind to deal only with the information that is task relevant is connected with its possibility to process fuzzy information. By concentrating only on the task-relevant information, the amount of information the brain has to deal with is reduced to a manageable level".

Fuzzy logic is basically a multi-valued logic that allows intermediate values to be defined between conventional evaluations like yes/no, true/false, black/white, etc. Notions like rather warm or pretty cold can be formulated mathematically and algorithmically processed. In this way, an attempt is made to apply a more human-like way of thinking in the programming of computers ("soft" computing).
Fuzzy logic systems address the imprecision of the input and output variables by defining fuzzy numbers and fuzzy sets that can be expressed in linguistic variables (e.g. small, medium, and large).
Fuzzy rule-based approach to modeling is based on verbally formulated rules overlapped throughout the parameter space. They use numerical interpolation to handle complex non-linear relationships.
Fuzzy rules are linguistic IF-THEN- constructions that have the general form "IF A THEN B" where A and B are (collections of) propositions containing linguistic variables. A is called the premise and B is the consequence of the rule. In effect, the use of linguistic variables and fuzzy IF-THEN- rules exploits the tolerance for imprecision and uncertainty. In this respect, fuzzy logic mimics the crucial ability of the human mind to summarize data and focus on decision-relevant information.
A more explicit form, we can generate FRBs with one of the following three types of rules:

- Fuzzy rules with a class in the consequent. This kind of rules has the following structure:

R_k : if x_1 is A_l^k and … and x_N is A_N^k then Y is C_j

Where $x_1 \ldots x_N$ are the outstanding selected features for the classification problem. A_l^k ... A_N^k are linguistic labels used to discretize the continuous domain of the variables, and Y is the class C_j to which the pattern belongs.

- Fuzzy rules with a class and certainty degree in the consequent.

R_k : if x_1 is A_l^k and … and x_N is A_N^k then Y is C_j with r^k

Where, r^k is the certainty degree of the classification in the class C_j for a pattern belonging to the fuzzy subspace delimited by the antecedent. This degree can be determined by the ratio $\frac{S_j^k}{S^k}$. Where, considering the matching degree as the compatibility degree between the rule antecedent and the pattern feature values.
S_j^k is the sum of the matching degrees for the class C_j patterns belonging to the fuzzy region delimited by the antecedent, and is the sum of the matching degrees for all the patterns belonging to this fuzzy subspace, regardless its associated class.

- Fuzzy rules with certainty degree for all classes in the consequent.

R_k : if x_1 is A_l^k and … and x_N is A_N^k then $(r_1^k, ..., r_M^k)$

Where, r_1^k is the soundness degree for the rule k to predict the class C_j for a pattern belonging to the fuzzy region represented by the antecedent of the rule. This degree of certainty can be determined by the same ratio as the type b) rules.
A Fuzzy Reasoning Method (FRM) is an inference procedure that derives conclusions from a set of fuzzy if-then rules and a pattern. The power of fuzzy reasoning is that we can achieve a result even when we do not have an exact match between a system observation and the antecedents of the rules.

Here, we only explain a classical FRM (Maximum Matching) although there are various methods, which you can find in [Zadeh, 1992] and [Cordon et al., 1999].

Suppose that the RB is R={ R_1,…, R_L } and there are Lj rules in R that produce class Cj. Clearly, in an RB type c), Lj = *L* and *j* = 1,… M.

For a pattern E^t = (e_1^t, ..., e_N^t), the classical *fuzzy reasoning method* considers the rule with the highest combination between the matching degree of the pattern with the if-part and the certainty degree for the classes. It classifies E^t with the class that obtains the highest value. This procedure can be described in the following steps.

- Let $R^k(E^t)$ denote the strength of activation of the if-part of the rule *k* (matching degree). Usually $R^k(E^t)$ is obtained by applying a t-norm to the degree of satisfaction of the clauses (x_i is A_i^k)

$$R^k(E^t) = \mathrm{T}(\mu_{A_N^k}(e_1^t)), \ldots, (\mu_{A_N^k}(e_N^t)) \tag{7}$$

- Let h($R^k(E^t)$, r_j^k)denote the degree of association of the pattern with class C_j according to the rule *k*. This degree is obtained by applying a combination operator between $R^k(E^t)$ and r_j^k . For this operator, we can use the minimum and product ones or the product and the arithmetic means.

For each class C_j, the degree of association of the pattern with the class, Y_j , is calculated by

$$Y_j = \max_{k \notin L_j} h(R^k(E^t), r_j^k) \tag{8}$$

This degree of association is a soundness degree of the classification of the pattern E^t in class C_j.

- The classification for the pattern E^t is the class C_h such

$$Y_h = \max_{j=1,\ldots,M} Y_j \tag{9}$$

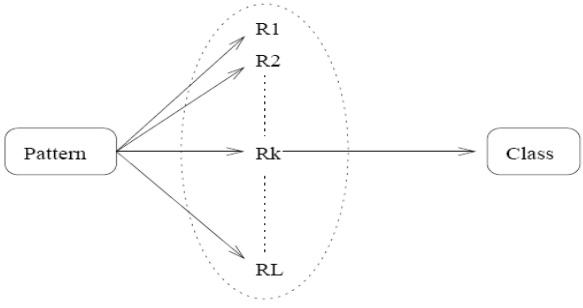


Figure2. Fuzzy Reasoning Method that uses only the winner rule

Graphically this method could be seen as shown in Figure 2. This reasoning method uses only one rule- the winner rule- in the inference process and wastes the information associated to all those rules whose association degree with the input pattern is lower than the association degree of the selected rule. If we only consider the rule with the highest association degree, we would not take into account the information of other rules with other fuzzy subsets which also have the value of the attribute in its support but to a lesser degree.

4. GENETIC FUZZY RULES BASED SYSTEM

Fuzzy systems have been successfully applied to problems in classification, modeling, control [Driankov et al. 1993], and in a considerable number of applications. In most cases, the key for success was the ability of fuzzy systems to incorporate human expert knowledge. In the 1990s, despite the previous successful history, the lack of learning capabilities characterizing most of the works in the field generated a certain interest for the study of fuzzy systems with added learning capabilities. Two of the most successful approaches have been the hybridization attempts made in the framework of soft computing with different techniques, such as neural and evolutionary; provide fuzzy systems with learning capabilities, as shown in Figure 3. Neuro-fuzzy systems are one of the most successful and visible directions of that effort. A different approach to hybridization leads to Genetic Fuzzy Systems (GFSs) [Cordon et al. 2004]

A GFS is basically a fuzzy system augmented by a learning process based on a GA. GA are search algorithms, based on natural genetics, that provide robust search capabilities in complex spaces, and thereby offer a valid approach to problems requiring efficient and effective search processes. Genetic learning processes cover different levels of complexity according to the structural changes produced by the algorithm, from the simplest case of parameter optimization to the highest level of complexity of learning the rule set of a rule based system. Parameter optimization has been the approach utilized to adapt a wide range of different fuzzy systems, as in genetic fuzzy clustering or genetic neuro-fuzzy systems.

Analysis of the literature shows that the most prominent types of GFSs are G*enetic Fuzzy Rule-Based Systems* (GFRBSs), whose genetic process learns or tunes different components of a Fuzzy Rule-Based System (FRBS). Figure 4 shows this conception of a system where genetic design and fuzzy processing are the two fundamental constituents. Inside GFRBSs, it is possible to distinguish between either parameter optimization or rule generation processes, that is, adaptation and learning.

A number of papers have been devoted to the automatic generation of the knowledge base of an FRBS using GA. The key point is to employ an evolutionary learning process to automate the design of the knowledge base, which can be considered as an optimization or search problem.

From the viewpoint of optimization , the task of finding an appropriate Knowledge Base (KB) for a particular problem, is equivalent to parameterize the fuzzy KB (rules and membership functions), and to find those parameter values that are optimal with respect to the design criteria. The KB parameters constitute the optimization space, which is transformed into a suitable genetic representation in which the search process operates.

The first step in designing a GFRBS is to decide which parts of the KB are subject to optimization by the GA. The KB of an FRBS does not constitute a homogeneous structure but is rather the union of qualitatively different components. As an example, the KB of a descriptive Mamdani-type FRBS (the one considered in Figure 5) is comprised of two components:A Data Base (DB), containing the definitions of the scaling functions of the variables and the membership functions of the fuzzy sets associated with the linguistic labels, and A Rule Base (RB), constituted by the collection of fuzzy rules.

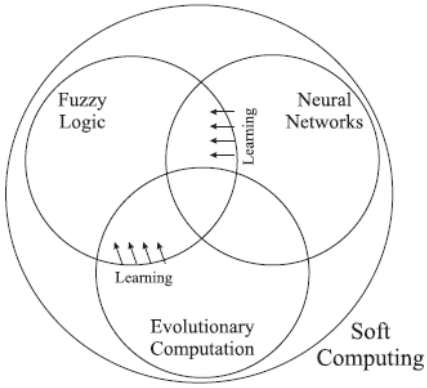


Figure 3. Soft computing and learning in fuzzy systems [Cordon et al. 2004]

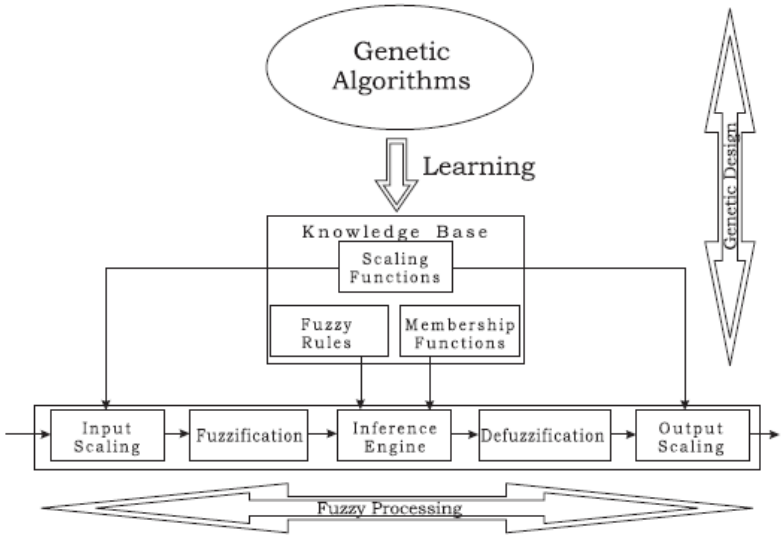


Figure4. Genetic design and fuzzy processing

The decision on which part of the KB to adapt depends on two conflicting objectives: dimensionality and efficiency of the search. A search space of a smaller dimension results in a faster and simpler learning process, but the obtainable solutions might be suboptimal. A larger, complete search space that comprises the entire KB and has a finer dimensionality is therefore more likely to contain optimal solutions, but the search process itself might become prohibitively inefficient and slow.
With these considerations there is an obvious trade-off between the completeness and dimensionality of the search space and the efficiency of the search. This trade-off offers different possibilities for GFS design.

First of all, it is important to distinguish between tuning (alternatively, adaptation) and learning problems:
Tuning is concerned with optimization of an existing FRBS, whereas learning constitutes an automated design method for fuzzy rule sets that starts from scratch. Tuning processes assume a predefined RB and have the objective to find a set of optimal parameters for the membership and/or the scaling functions, DB parameters.
Learning processes perform a more elaborated search in the space of possible RBs or whole KB and do not depend on a predefined set of rules.

4.1. Genetic tuning

Tuning of the scaling functions and fuzzy membership functions is an important task in FRBS design. Parameterized scaling functions and membership functions are adapted by the GA according to fitness function that specifies the design criteria in a quantitative manner.
As previously said tuning processes assume a predefined RB and have the objective of finding a set of optimal parameters for the membership and/or the scaling functions (Figure 5). It is also possible, as will be seen, to perform the tuning process a priori, i.e., considering that a subsequent process will derive the RB once the DB has been obtained, that is a priori genetic DB learning.

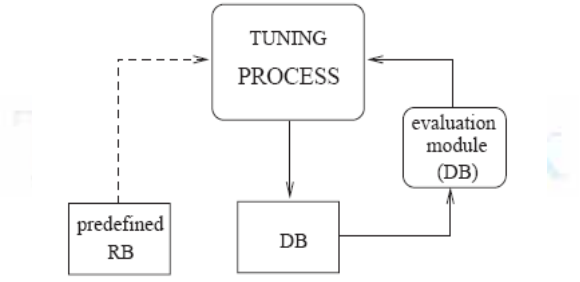


Figure 6. Tuning the data base

Tuning scaling functions
Scaling functions applied to the input and output variables of FRBSs normalize the universes of discourse in which the fuzzy membership functions are defined. Usually, the scaling functions are parameterized by a single scaling factor [Ng and Li, 1994] or a lower and upper bound in case of linear scaling, and one or several contraction/dilation parameters in case of non-linear scaling. These parameters are adapted such that the scaled universe of discourse better matches the underlying variable range.
The usual approach of these kinds of processes is the adaptation of one to four parameters (defining the scaling function) per variable: one when using a scaling factor, two for linear scaling, and three or four in non-linear scaling. Most of the cited works consider a real coding scheme to represent the parameters of the function, but it is also possible to use binary codes, as in [Ng and Li, 1994] where a three bits binary representation of each scaling factor is used.
Since the number of variables is predefined, as well as the number of parameters required to code each scaling function, this approach leads to a fixed length code.

Tuning membership functions

When tuning membership functions, an individual represents the entire DB as its chromosome encodes the parameterized membership functions associated to the linguistic terms in every fuzzy partition considered by the FRBS. The most common shapes for the membership functions (in GFRBSs) are triangular (either isosceles or asymmetric), trapezoidal or Gaussian functions. The number of parameters per membership function usually ranges from one to four, each parameter being either binary or real coded.

The structure of the chromosome is different for FRBSs of the descriptive (using linguistic variables) or the approximate (using fuzzy variables) type. When tuning the membership functions in a linguistic model, the entire fuzzy partitions are encoded into the chromosome and it is globally adapted to maintain the global semantic in the RB. These approaches usually consider a predefined number of linguistic terms for each variable (no need to be the same for each of them), leading to a code of fixed length in what concerns membership functions. But even having a fixed length for the code, it is possible to evolve the number of linguistic terms associated to a variable by simply defining a maximum number (that defines the length of the code) and letting some of the membership functions to be located out of the range of the linguistic variable (reducing the actual number of linguistic terms). This is the conception that is used when designing a TSK system with linguistic input variables.

A particular case, where the number of parameters to be coded is reduced, is that of descriptive fuzzy systems working with strong fuzzy partitions. In this case, the number of parameters to code is reduced to those defining the core regions of the fuzzy sets: the modal point for triangles, the extreme points of the core for trapezoidal shapes.

On the other hand, tuning the membership functions of a model working with fuzzy variables (scatter partitions) is a particular instantiation of KB learning since the rules are completely defined by their own membership functions instead of referring to linguistic terms in the DB.

4.2. Genetic learning of rule bases

Genetic learning of RBs assumes a predefined set of fuzzy membership functions in the DB to which the rules refer to by means of linguistic labels (Figure 6). It only applies to descriptive FRBSs, as in the approximate approach adapting rules, which is equivalent to modify the membership functions.

When considering a rule based system and focusing on learning rules, there are three main approaches that have been applied in the literature: Pittsburgh [Smith, 1980], Michigan [Holland and Reitman, 1978], and iterative rule learning [Venturini, 1993]. Pittsburgh and Michigan approaches are the most extended methods for rule learning developed in the field of GA. The first one is characterized by representing an entire rule set as a genetic code (chromosome), maintaining a population of candidate rule sets and using selection and genetic operators to produce new generations of rule sets. The Michigan approach considers a different model where the members of the population are individual rules and a rule set is represented by the entire population. In the third approach, the iterative one,

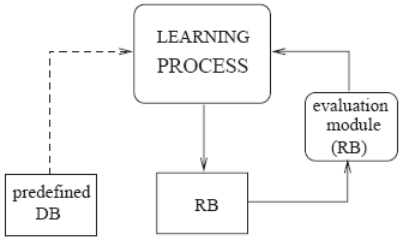


Figure 7. Learning the rule base

chromosomes code individual rules, and a new rule is adapted and added to the rule set, in an iterative fashion, in every run of the GA.

The three learning approaches described can be considered to learn RBs. The RB can be represented by a relational matrix, a decision table, or a list of rules.

Representations through relational matrix and decision table are only useful when the system has a reduced number of variables, since they lead to an unaffordable length of the code when having more than two or three input variables. The result is a monolithic code that can be only managed by the Pittsburgh approach.

The list of rules is the most used representation, which adopts quite different codes for the individual rules, and can be adapted to the three learning approaches. Often the number of rules in the list is variable (having in some cases an upper limit). A common approach to code individual rules is the use of the Disjunctive Normal Form (DNF) represented in the form of a fixed length binary string. DNF rules are also considered when working with variable length codes based on messy GA [Mitchell, 1998]. With a structure of list of rules, the chromosome can be generated by concatenating the code of individual rules (Pittsburgh, where each chromosome codes a RB) or will contain the code of a single rule.

To code a rule, either as an element of an RB that generates a chromosome or as a single rule generating a chromosome by itself, there are many different approaches. Rules are composed of propositions of the form *variable* is *value*, where the variable could be identified by position or by label, and the value could have quite different forms (Figure 7). When using a code with positional structure (Figure 7, top) there is a fixed location for each variable in which the information (*value*) related to that variable is placed. When using non-positional codes (Figure 7, bottom), the code of a rule is composed of pairs (*var* , *value*), where *var* is a label identifying the variable. In both cases, positional and non-positional codes, the content of the value part can be: the label of a linguistic term (linguistic variables), the binary code of a DNF structure (linguistic variables), the parameters defining a fuzzy set (fuzzy variables), or the real values (coefficients) of the linear output (output variables of TSK rules).

In addition to the RB learning, other approaches try to improve the preliminary DB definition once the RB has been obtained. That process is comprised by a learning process to obtain the RB considering a predefined DB, followed by a learning process similar to those described before. In this case, the tuning process that involves the DB learning is called a posteriori DB learning. Figure 8 shows this approach.

Genetic learning of knowledge bases

Since genetic learning of the KB deals with heterogeneous search spaces (Figure 9), it encompasses different genetic representations such as variable length chromosomes, multi-chromosome genomes and chromosomes encoding single rules instead of a whole KB. The computational cost of the genetic search grows with the increasing complexity of the search space. A GFRBS that encodes individual rules rather than entire KBs is an option to maintain a flexible, complex rule space in which the search for a solution remains feasible and efficient. Again, the three learning approaches can considered: Michigan, Pittsburgh, and iterative rule learning approach.

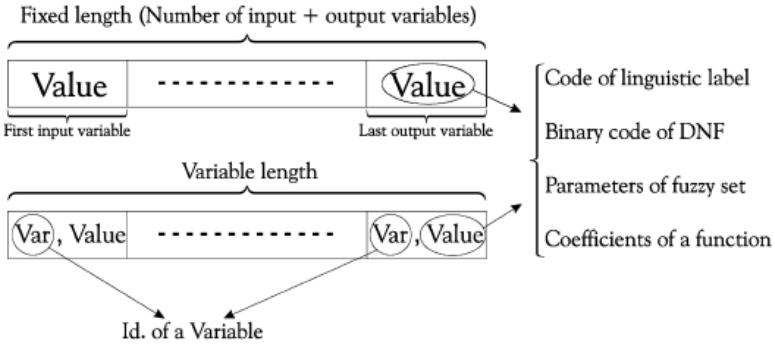


Figure 9. Representing a rule with fixed or variable length codes

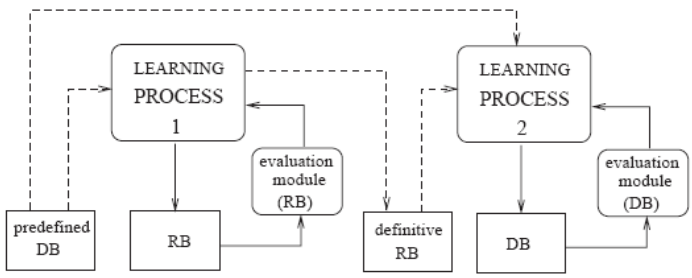


Figure 8. Learning the rule base and, a posteriori, the data base

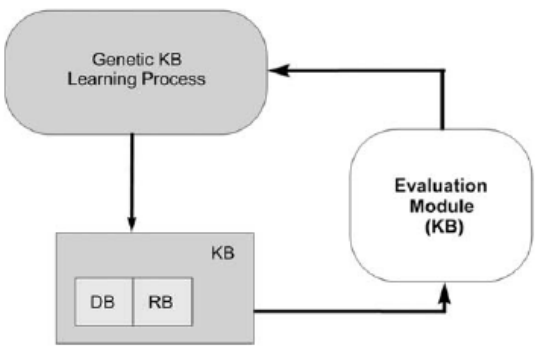


Figure 9. Learning the knowledge base

Following, we describe the four approaches that canbe found within the genetic learning of a KB.

1. *Genetic rule learning*. Most of the approaches proposed to automatically learn the KB from numerical information have focused on the RB learning, using a predefined DB. The usual

way to define this DB involves choosing a number of linguistic terms for each linguistic variable (an odd number between 3 and 9, which is usually the same for all the variables) and setting the values of the system parameters by a uniform distribution of the linguistic terms into the variable universe of discourse. Figure 10 shows graphically this type of RB learning.

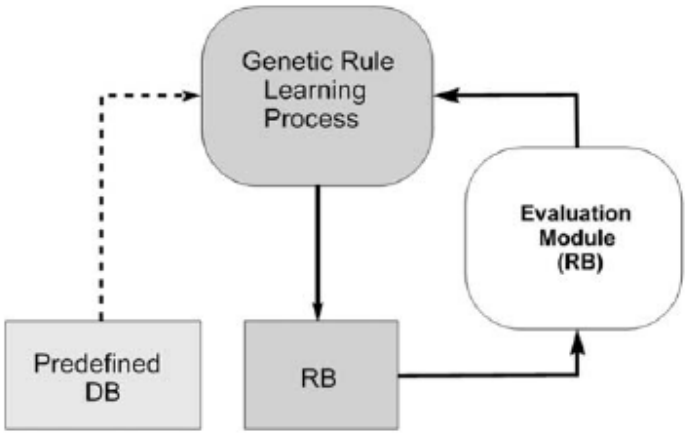


Figure 10. Genetic rule learning process

1. *Genetic rule selection*. Sometimes we have a big number of rules extracted via a data mining method that only provide us a big number of rules associated to our problem. A big RB and an excessive number of rules makes difficult to understand the FRBS behavior. Thus we can find different kinds of rules in a fuzzy rule set: irrelevant rules, redundant rules, erroneous rules and conflictive rules, which perturb the FRBS performance when they coexist with others. To face this problem we can use a genetic rule selection process for obtaining an optimized subset of rules from a previous fuzzy rule set by selecting some of them. Figure 1 graphically shows this idea.
2. *Genetic DB learning*. There is another way to generate the whole KB that considers two different processes to derive both components, DB and RB. A DB generation process allows us to learn the shape or the membership functions and other DB components such as the scaling functions, the granularity of the fuzzy

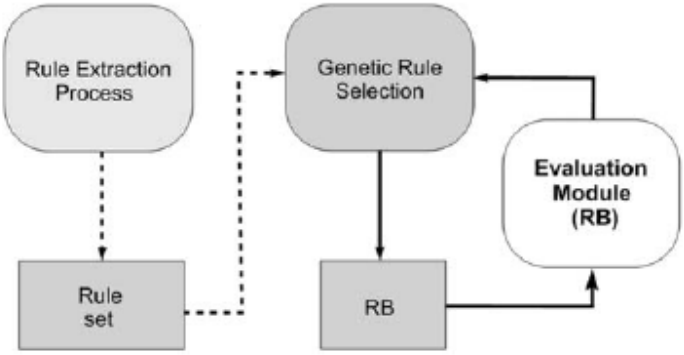


Figure 11. Genetic rule selection process

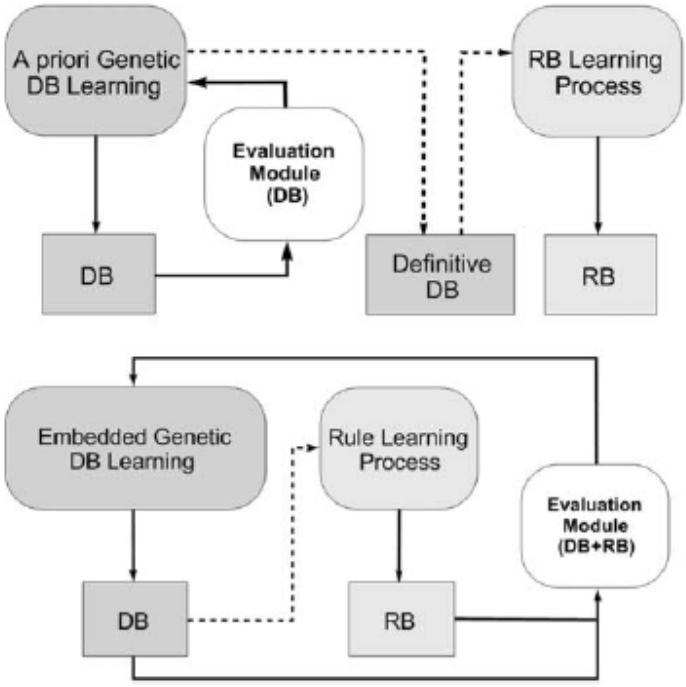


Figure 12. Genetic DB learning (embedded and a priori)

partitions, ... This DB generation process can use a measure for evaluating the quality of the DB, we can call them as ''A priori genetic DB learning''. The second possibility is to consider and embedded genetic learning process where the DB generation process wraps an RB learning one working as follows: each time a DB has been obtained by the DB definition process, the RB generation method is used to derive the rules, and some type of error measure is used to validate the whole KB obtained. We should note this operation mode involves a partitioning of the KB learning problem. These two kinds of learning models are represented in Figure 12

3. *Simultaneous genetic learning of KB components*. Other approaches try to learn the two components of the KB simultaneously. This kind of learning is depicted in Figure 11. Working in this way, they have the possibility of generating better definitions but there is a need to deal with a larger search space that makes the learning process more difficult and slow.

5. LEARNING FUZZY RULES USING ANT COLONY OPTIMIZATION ALGORITHM

Here we discuss a method of facing the Fuzzy Rule Learning problem making use of ACO algorithms. To do so, the FRL problem will be formulated as an optimization problem and the features related to these kinds of algorithms—such as heuristic information, pheromone initialization, fitness function, solution construction, and pheromone update—will be introduced [Casillas et al. 2000]. To apply ACO algorithms to a specific problem, the following steps have to be performed:

- Obtain a problem representation as a graph or a similar structure easily covered by ants.
- Define the way of assigning a heuristic preference to each choice that the ant has to take in each step to generate the solution.

- Establish an appropriate way of initializing the pheromone.
- Define a fitness function to be optimized.
- Select an ACO algorithm and apply it to the problem.

In the following subsections, these steps will be introduced to solve the FRL problem.

Problem Representation

To apply ACO algorithms to the FRL problem, it is convenient to see it as a combinatorial optimization problem with the capability of being represented on a graph. In this way, we can face the problem considering a fixed number of rules and interpreting the FRL problem as the way of assigning consequents (i.e., labels of the output fuzzy partition) to these rules with respect to an optimality criterion [Casillas et al. 2000].

Hence, we are in fact dealing with an assignment problem and the problem representation can be similar to the one used to solve the Quadratic Assignment Problem (QAP) [Bonabeau et al. 1999], but with some peculiarities. We may draw an analogy between *rules* and facilities and between *consequent* and locations. However, unlike the QAP, the *set of possible consequents for each rule may be different* and *it is possible to assign a consequent to more than one rule* (two rules may have the same consequent). We can deduce from these characteristics that the order of selecting each rule to be assigned a consequent is not determined, i.e., the assignment order is irrelevant.

To construct the graph, the following steps are taken:

Determine the rules: A rule R_i and i=1,…,N_r defined by an antecedent combination,

R_i : IF X_1 is A_{i1} and … and X_n is A_{in} , Will take part in the graph if and only if:

$\exists e_l = (x_1^l,...,x_n^l,y^l) \in$ E such that $\mu_{A_{i1}}(x_1^l)\cdot...\cdot\mu_{A_{in}}(x_n^l) \neq 0$

That is, there is at least one example located in the fuzzy input subspace defined by the antecedents considered in the rule.

Link the rules to consequents: The rule R_i will be linked to the consequent Bj and j = 1,…, Nc , (taken from the set of labels of the output fuzzy partition) if and only if it meets the following condition:

$\exists e_l = (x_1^l,...,x_n^l,y^l) \in$ E such that $\mu_{A_{i1}}(x_1^l)\cdot...\cdot\mu_{A_{in}}(x_n^l)\cdot\mu_{B_j}(y^l) \neq 0$

That is, there is at least one example located in the fuzzy input subspace that is covered by such a consequent.

Figure 12 shows an example of a system with four rules and one output variable with three consequents. In Figure 12(a), the possible consequents for each antecedent combination are shown. To construct a complete solution, an ant iteratively goes over each rule and chooses a consequent with a probability that depends on the pheromone trail τ_{ij} and the heuristic information η_{ij} , as usual (see Figure 12(b)). As said, the order of selecting the rules is irrelevant. In Figure 12(c) we may see the possible paths that an ant can take in a specific example.

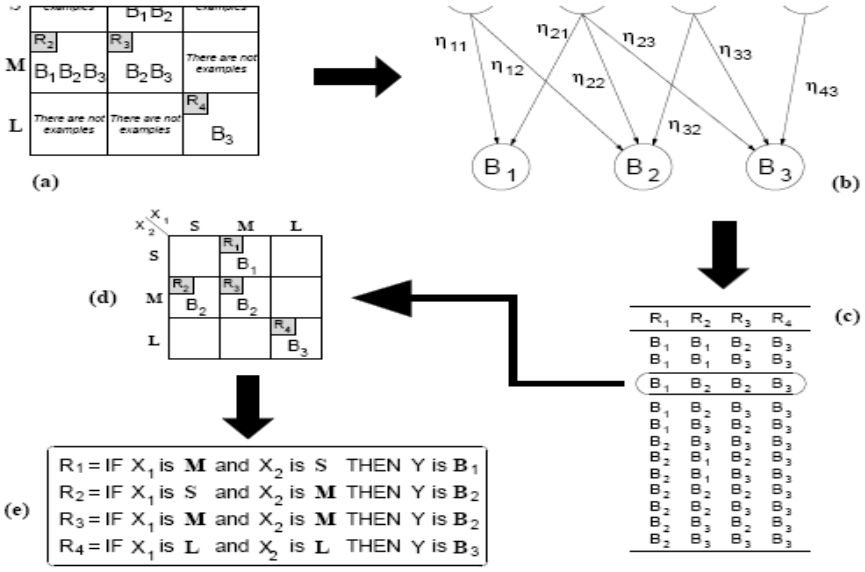


Figure 12. Learning process for a simple problem with two input variables (n = 2), four rules (Nr = 4), and three labels in the output fuzzy partition (Nc=3): (a) Set of possible consequent for each rule (only the rules where at least one example is located in the corresponding subspace are considered); (b) Graph of paths where ηij≠0 except η_{13}, η_{31}, η_{41}, and η_{42}, which are zero; (c) It is possible to take twelve different paths (combinations of consequents); (d) Rule decision table for the third combination; (e) RB generated from the third combination [Bonabeau et al. 1999].

Heuristic Information

The heuristic information on the potential preference of selecting a specific consequent, B_j, in each antecedent combination (rule) is determined by considering covering criteria as follows (see Figure 13 for a graphical interpretation of the heuristic assignment) [Bonabeau et al. 1999]:
For each rule defined by an antecedent combination,

R_i = IF X_1 is A_{i1} and … and X_n is A_{in} where i = 1, …, N_r do:

Build the set E'_i composed of the input-output data pairs that are located in the input subspace defined by R_i, i.e.,

$$E'_i = \{e_l = (x_1^l,...,x_n^l;y^l) \in E \text{ such that } \mu_{A_{i1}}(x_1^l)\cdot...\cdot\mu_A\}$$

Make use of an initialization function based on covering criteria to give a heuristic preference degree to each election. Many different choices may be considered. Here, we discuss the covering of the example best covered criterion shown in Figure 13.
Since the heuristic information is based on covering criteria, it will be zero for a specific consequent when no examples located in the fuzzy input subspace are covered by it. This means that for a rule, only those links to consequents whose heuristic information is greater than zero will be considered. In Figure 12(b), we can observe the consequent B_3 cannot be assigned to the rule R_1, the consequent B_1

cannot be assigned to the rule R_3, and the consequents B_1 and B_2 cannot be assigned to the rule R_4 because their heuristic information (covering degrees) are zero.

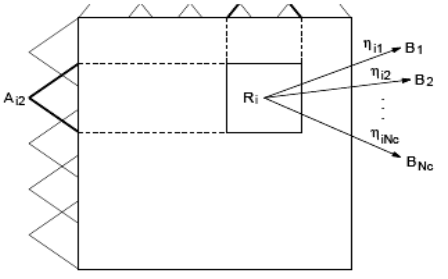
Pheromone Initialization

The initial pheromone value of each assignment is obtained as follows:

$\tau_0 = \frac{\sum_{i=1}^{N_r} \max_{j=1}^{N_c} \eta_{ij}}{N_r}$. In this way, the initial pheromone will be the mean value of the path constructed taking the best consequent in each rule according to the heuristic information (a greedy assignment).

Fitness Function

The fitness function establishes the quality of a solution. The measure considered will be the function called *Mean Square Error* (MSE), which defined as $\mathrm{MSE}(\mathrm{RB_k}) = \frac{1}{2.|E|}\sum_{e_j \in E}(y^l - F_k(x_0^l))^2$, with $F_k(x_0^l)$ being the output obtained from the FRBS (built using the RB generated by the ant k, RB_k) when receives the input x_0^l (input component of the example e_l), and y^l being the known desired output. The closer to zero the measure is, the better the solution is.



Heuristic information considered:

$$\eta_{ij} = \max_{e_l \in E'_i} Min\,(\mu_{A_{i1}}(x_1^l), \ldots, \mu_{A_{in}}(x_n^l), \mu_{B_j}(y^l))$$

Figure 13. Heuristic assignment from the rule *Ri* to each consequent in a system with two input variables, five labels for each of them, and *Nc* labels (consequents) in the output fuzzy partition. The *covering of the example best covered* is considered to be the heuristic information.

Ant Colony Optimization Algorithm

Once the previous components have been defined, an ACO algorithm has to be given to solve the problem. In this contribution, two well-known ACO algorithms will be considered: the Ant System (AS) and the Ant Colony System (ACS). Depending on the ACO algorithm followed, two methods arise: the AS-FRL and the ACS-FRL ones. The so-known *solution construction* and *pheromone trail update rule* considered by these ACO algorithms will be used.

Only some adaptations will be needed to apply them to the FRL problem:

- The set of nodes attainable from R_i will be $J_k(i) = \{j \ \textit{such that}\ \eta_{ij} \neq 0\}$ in the transition rules considered by both ACO algorithms when constructing the solution.

- The amount of pheromone ant *k* puts on the couplings belonging to the solution constructed by it will be $1/MSE(RB_k)$, with RB_k being the RB generated by ant *k*.
- In the *local pheromone trail update rule* of the ACS algorithm, the most usual way of calculating $\Delta\tau_{ij}, \Delta\tau_{ij} = \tau_0$, will be used, thus considering the simple-ACS algorithm.

6. APPLICATION OF PSO FOR FUZZY RULE BASED SYSTEM

A structure of the weighted fuzzy rule-based system is proposed in Figure 14. The output is obtained by the average of the "then" of rules. For a particular instance $x = \{x_1, x_2, \ldots, x_n\}$, output y=F(x) can be further calculated using [Liu et al. 2004]:

$$y = F(x) = \frac{\sum_{i=1}^{m} \omega_i \times \mu_{ik} \times c_i}{\sum_{i=1}^{m} \omega_i \times \mu_{ik}} \tag{10}$$

Where ω_i is the weight of rule and $\mu_{Ri}(x) = \min \mu_{R_{n1}}(x_n)$, which denotes the degree that instance x matches the rule $R_i, c_i \in \{C_1, \ldots, C_M\}$ is the centroid of every fuzzy set C_i.

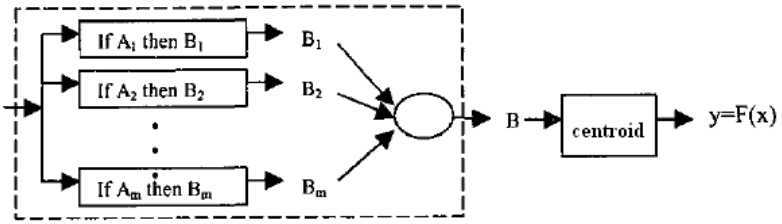


Figure 14. Weighted fuzzy rule-based system [Liu et al. 2004]

Here, to obtain the parameters of the function of the weighted fuzzy system described in Eq. 10, the PSO algorithm is used to learn the parameters based on the data. The implementation procedures are as following [Liu et al. 2004]:

Step 1: Initialize related parameters, including the size of swarm m, the inertia weight w, the acceleration constants c_1 and c_2, the maximum velocity $V_{\max}$, the stop criterion and the *m* particles of a population, which include the random location X1 in the problem space and the velocity for the dth dimension.

Step 2: Evaluate the desired optimization fitness function for each particle.

Step 3: Compare the evaluated fitness value of each particle with its P_{best} . If current value is better than P_{best} then set the current location as the P_{best} location. Furthermore, if current value is better than g_{best} , then reset g_{best} to the current index in particle array.

Step 4: Change the velocity and location of the particle according to the Eqs. 11a and 11b, respectively.

$$V_{id}^{n+1} = wv_{id}^{n} + c_1 r_1^{n}(p_{id}^{n} - x_{id}^{n}) + c_2 r_2^{n}(p_{id}^{n} - x_{id}^{n}) \tag{11a}$$

$$X_{id}^{n+1} = x_{id}^{n} + v_{id}^{n+1} \tag{11b}$$

Step 5: Loop to step 2 until a stopping criterion is met. The criterion usually is a sufficiently good fitness value or a predefined maximum number of generations Gmax. The fitness function mentioned in step 2 and step 3 for the weighted fuzzy rule-based system identification is given as:

$$J(t) = \sum_{k=1}^{n} [y(k) - \hat{y}(k)]^2 \tag{12}$$

Where J = cost function to be minimized, y = system output, $\hat{y}$ = model output, and k= 1, 2, …, n. n denotes the number of sample data.

Parameters Learning by PSO Algorithm

Before PSO parameter learning stage, the initial feature of the population for PSO should be defined. This can be done for example via cluster centers driven by popular FCM. These initial values will sequentially be assigned to the premise (a_i) and the consequent part (y_i) of the fuzzy set. Normally, the parameter set (b_i) is determined by random operation. Note that every particle's position is made up of parameter set (a_{ij} , b_{ij} and y_i) and its velocity is initiated as same dimension as position by random process. At the learning iteration, each particle's position (*px*) and velocity (*pv*) values are adjusted by two best information. The first best one, called *Pbest_px*, is every particle's best solution (highest fitness) which has been achieved so far. The other best value, called *Gbest_px*, is obtained by choosing the overall best value from all particles. It is known that this useful information is good for all swarms to share their best knowledge. At the learning step, the velocity of every particle is modified according to the relative *Pbest_px* and *Gbest_px*. The new velocity is updated by the following equation[Feng H.M.; 2006]:

$$pv_{p,d}(t+1) = pv_{p,d}(t) + \alpha 1_{p,d}(t+1)(Pbest\_px_{p,d}(t+1) - px_{p,d}(t)) + \alpha 2_{p,d}(t+1)$$
$$(Gbest\_px_{p,d}(t+1) - px_{p,d}(t)), 1 \le d \le n \tag{13}$$

Where $pv_{p,d}$ and $px_{p,d}$ are respectively the *p*th velocity and position. Here, *p* is the number of particles; *d* the number of dimension. t employs current state, t + 1 denotes the next time step, $\alpha 1_{p,d}$ *(t + 1)* and $\alpha 2_{p,d}(t+1)$ are random number between 0 and 1.
Since the velocity of the particle is determined, the next particle's location will be modified by:

$$px_{p,d}(t+1) = px_{p,d}(t) + pv_{p,d}(t+1) \tag{14}$$

Based on (13) and (14) the direction of every particle will regulate its original flying path with the factor of *Gbest_px* and *Pbest_px*.

7. IMMINENT TRENDS

The hybridization of fuzzy logic and EC in Genetic Fuzzy Systems became an important research area during the last decade. To date, several hundred papers, special issues of different journals and books have been devoted [cordon et al. 2001], [Cordon, et al. 2004].
Nowadays, as the field of GFSs matures and grows in visibility, there is an increasing concern on the integration of these two topics from a novel and more sophisticated perspective.
In what follows, we enumerate some open questions and scientific problems that suggest future research:
Hybrid intelligent systems balance between interpretability and accuracy, and sophisticated evolutionary algorithms. As David Goldberg stated [Goldberg, 2004], the integration of single methods into hybrid intelligent systems goes beyond simple combinations [Cordon et al., 2004].
Other evolutionary learning models, as seen, the three classical genetic learning approaches are Michigan, Pittsburgh and Iterative Rule Learning. Improving mechanisms for these approaches or the development of other new ones may be necessary [Cordon et al., 2004].

8. CONCLUSION

In this chapter we focused on three bio-inspired algorithms and their combination with fuzzy rule based systems. Rule Based systems are widely being used in decision making, control systems and forecasting. In the real world, much of the knowledge is imprecise and ambiguous but fuzzy logic provides our systems a better presentation of the knowledge, which is in terms of the natural language. Fuzzy rule based systems constitute an extension of classical Rule-Based Systems, because they deal with fuzzy rules instead of classical logic rules.
As we said, the generation of the Knowledge Base (KB) of a Fuzzy Rule Based System (FRBS) presents several difficulties because the KB depends on the concrete application, and this makes the accuracy of the FRBS directly depend on its composition. Many approaches have been proposed to automatically learn the KB from numerical information. Most of them have focused on the Rule Base (RB) learning, using a predefined data base (DB). This operation mode makes the DB have a significant influence on the FRBS performance. In fact, some studies have shown that the system performance is much more sensitive to the choice of the semantics in the DB than to the composition of the RB. The usual solution for improving the FRBS performance by dealing with the DB components involves a tuning process of the preliminary DB definition once the RB has been derived. This process only adjusts the membership function definitions and does not modify the number of linguistic terms in each fuzzy partition since the RB remains unchanged.

Using bio-inspired optimization algorithms have been shown effective in improving the tuning the parameters and learning the Fuzzy Rule Base. Thus, we began by introducing the EC and Swarm Intelligence as a subfield of Computational Intelligence that involves Combinatorial Optimization problems and then three following bio-inspired algorithms, i.e. GA, Ant Colony, and PSO were explained. We also represented the usage of these algorithms in improving the Knowledge Based systems with focus on Rule Base Learning. In this context we emphasized the need
to build hybrid intelligent systems that go beyond simple combinations. Development of GFSs that offer acceptable trade-off between interpretability and accuracy is also a major requirement for efficient and transparent knowledge extraction.
The hybridization between fuzzy systems and GAs in GFSs became an important research area during the last decade. GAs allow us to represent different kinds of structures, such as weights, features together with rule parameters, etc., allowing us to code multiple models of knowledge representation. This provides a wide variety of approaches where it is necessary to design specific genetic components for evolving a specific representation. Nowadays, it is a mature research area, where researchers need to reflect in order to advance towards strengths and distinctive features of the GFSs, providing useful advances in the fuzzy systems theory.
In this chapter also, an interesting application, the FRL problem (which involves automatically learning from numerical data the RB composing an FRBS), has been proposed to be solved by the ACO meta-heuristic. In this way, an ACO-based learning method has been presented. Comparing with other ad hoc learning algorithms, the models obtained by the ACO methods are clearly better. Moreover, opposite to other kinds of optimization techniques such as SA and GA, the ACO approach performs a quick convergence and sometimes obtains better results. The former is due to the use of heuristic information to guide the global search. Then, as it was shown, based on the fuzzy rule-based system, we can assign a weight to every fuzzy rule to construct a weighted fuzzy rule-based system. The PSO algorithm can be used to estimate the parameters of the weighted fuzzy rule-based system, which include the position and the shape of membership function, fuzzy rules and its weights. The PSO algorithm is effective in the procedure of obtaining the parameters of the weighted fuzzy system. As we said, fuzzy system with weights could get better results compared with the one without weights.

REFERENCES

Back T., Schwefel H.P. [1986] An overview of Evolutionary Algorithm for Parameter Optimization, Springer Verlag, New York-1986

Bárdossy A., Duckstein L.; [1995]; Fuzzy rule-based modeling with applications to geophysical; CRC Press.

Bonabeau E., Dorigo M., and Theraulaz G.; [1999]; swarm intelligence from natural to artificial systems; Oxford University Press.

Bonabeau E., Dorigo M., Theraulaz G.; [2000]; Nature; Vol. 406; 6791, 39-42.

Driankov D., Hellendoorn H., Reinfrank M.;[1993]; An Introduction to Fuzzy Control; Springer, Berlin.

Casillas J., Cordon O., Herrera F.; [1999]; Learning Fuzzy Rules Using Ant Colony Optimization Algorithms; Ant Colonies to Arti Ants: A Series of International Workshops on Ant Algorithms;

Chi Z., Yan H., and Pham T.; [1996]; Fuzzy algorithms: With applications to image processing and pattern recognition; Advances in Fuzzy Systems-Application and Theory; Vol. 10; World Scientific.

Cordon, O., Del Jesus, M.J. and Herrera, F.; [1999]; A proposal on reasoning methods in fuzzy rule-based classification systems; International Journal of Approximate reasoning. 20. 21-45.

Cordon O, Gomide F., Herrera F., Hoffmann L., Magdalena L.;[2004];Ten years of genetic fuzzy systems: current framework and new trends; Fuzzy Sets and Systems; 141; 5–31;

Cordón O., Herrera F., and Villar P.; [1998]; Generating the Knowledge Base of a Fuzzy Rule-Based System by the Genetic Learning of the Data Base; International journal of intelligent systems; **13**, 1025-1053.

Cordon O., Herrera F., Hoffmann F., and Magdalena L.; [2001]; Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases; Vol. 16; World Scientific (Advances in Fuzzy Systems-Application and Theory).

Dorigo M.; [1992]; Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italie.

Dorigo M. and Gambardella L. M.; [1997]; Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transaction on Evolutionary Computation; 1;53--66.

Eiben A. E. and Smith J. E.; [2003]; introduction to evolutionary computing; Springer,Natural Computing Series.

Engelbrecht P.; [2007]; Computational Intelligence: An Introduction, 2nd Edition; Wiley.

Feng H.F.; [2006]; Hybrid Stages Particle Swarm Optimization Learning Fuzzy Modeling Systems Design; Journal of Science and Engineering, Vol. 9, No 2, pp. 167-176

Fisher, R. A.; [1911]; Mendelism and biometr; Unpublished

Gambardella L.M. and M. Dorigo M.; [1995]; Ant-Q: a reinforcement learning approach to the travelling salesman problem, Proceedings of the Twelfth International Conference on Machine Learning, ML-95, Palo Alto, CA, Morgan Kaufmann, Palo Alto: California, USA,.

Goldberg D. E.; [1989]; Genetic algorithms in search, optimization, and machine learning; Addison-Wesley: New York.

Goldberg D. E.; [2000]; A meditation on the computational intelligence and its future, Technical Report ; Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Foreword of the Proc. Int. Symp. on Computational Intelligence.

Holland J.H., Reitman J.S.; [1978], Cognitive systems based on adaptive algorithms, D.A. Waterman, F. Hayes-Roth (Eds.), Pattern-Directed Inference Systems, Academic Press, New York, 313–329.

Kennedy J., Eberhart R.C.; [1995]; Particle Swarm Optimization, Proceedings of IEEE international conference on neural networks, IV, pages 1942--1948, Piscataway, NJ, 1995. IEEE Service Center

Kennedy J. and Eberhart R. C.; [2001]; Swarm Intelligence; Morgan Kaufmann Publishers: San Francisco.

Klir, G. and Yuan, B.; [1995]; Sets and Fuzzy Logic, Theory and Applications; Prentice Hall.

Leondes T; [1999]; Fuzzy Theory Systems: Techniques and Applications; Academic Press, Inc.

Lin S. and Kernighan B.W.; [1973]; An effective heuristic algorithm for the traveling salesman Problem; Operations Research, 21, 498–516.

L.M Gambardella and M. Dorigo M.; [1996]; Solving Symmetric and Asymmetric TSPs by Ant Colonies; Proceedings of the IEEE Conference on Evolutionary Computation, ICEC96, Nagoya, Japan, 622-627.

Mamdani E.H. ; [1982]; A fuzzy rule-based method of controlling dynamic processes, IEEE Xplore Volume: 20, pp. 1098-1103

Maniezzo V. ,Gambardella L. M., Luigi F. D.; [2004]; Optimization Techniques in Engineering; Chapter 5; Springer-Verlag.

Mitchell M.; [1998]; An introduction to genetic algorithm; The MIT Press.

Mitchell T.; [1997]; Machine Learning, International Edition: USA, McGrawHill.

Ng K.C., Li Y.; [1994]; Design of sophisticated fuzzy logic controllers using genetic algorithms, Proc. 3rd IEEE Int.Conf. on Fuzzy Systems (FUZZ-IEEE'94), 3, Orlando, FL, USA, 1708–1712.

Pedrycz W.; [1996]; Knowledge-based clustering: from data to information granules; Wiley «a».

Pedrycz W.; [1996]; Fuzzy Modeling: Paradigms and Practices; Kluwer Academic Publishers «b».

Smith S.F.; [1980]; A learning system based on genetic adaptive algorithms; Doctoral Dissertation, Department of Computer Science, University of Pittsburgh.

Venturini G.; [1993]; SIA: a supervised inductive algorithm with genetic search for learning attribute based concepts, In Proc. European Conf. on Machine Learning, Vienna, 280–296.

Yager R., Zadeh L. A.; [1992]; An Introduction to Fuzzy Logic Applications in Intelligent Systems; Kluwer Academic Publishers.

Yijian Liu, Xuemei Zhu, Jianming Zhang, Shuqing Wang; [2004]; Application of Particle Swarm Optimization Algorithm for Weighted Fuzzy Rule-Based System;30th Annual Conference of the IEEE Industrial Electronics Society, Ousan, Korea.

Yoshinari Y., Pedrycz W., and Hirota K.; [1993]; Construction of fuzzy models through clustering techniques; Fuzzy Sets and Systems; Vol. 54; Issue 2;157 - 165; Elsevier North-Holland, Inc.

Zadeh, L.A.; [1992]; The calculus of fuzzy if-then rules; AI Expert. Vol.7; 3, 23-27.

Zimmermann H.J. ; [1996]; Practical applications of fuzzy technologies; Springer.

TMRF e-Book
Advanced Knowledge Based Systems:
Model, Applications & Research
(Eds. Sajja & Akerkar), Vol. 1, pp 160 – 187, 2010

Chapter 8

Structure-Specified Real Coded Genetic Algorithms with Applications

Chun-Liang Lin*, Ching-Huei Huang and Chih-Wei Tsai

Absract This chapter intends to present a brief review of genetic search algorithms and introduce a new type of genetic algorithms (GAs) called the real coded structural genetic algorithm (RSGA) for function optimization. The new genetic model combines the advantages of traditional real genetic algorithm (RGA) with structured genetic algorithm (SGA). This specific feature makes it able to solve more complex problems efficiently than that has been made through simple GAs. The fundamental genetic operation of the RSGA is explained and major applications of the RSGA, involving optimal digital filter designs and advanced control designs, are introduced. Effectiveness of the algorithm is verified via extensive numerical studies.

1. INTRODUCTION

Recently, new paradigms based on evolutionary computation have emerged to replace the traditional, mathematically based approaches to optimization (Ashlock 2006). The most powerful of these is genetic algorithm (GA), which is inspired by natural selection, and genetic programming. Since GAs were first introduced by Holland in the 1970s, this method has been widely applied to different engineering fields in solving for function optimization problems (Holland 1975). Conventional binary genetic algorithms (BGAs), which emphasize the use of binary coding of the chromosomes, is satisfactory in solving various problems. However, it requires an excessive amount of computation time when dealing with higher dimensional problems that require higher degree of precision. To compensate for the excessive computation time required by the BGA, the real genetic algorithm (RGA) emphasizing on the coding of the chromosomes with floating point representation was introduced and proven to have significant improvements on the computation speed and precision (Gen and Cheng 2000; Goldberg 1991). At the same time, much effort was imposed to improve computation performance of GAs and avoid premature convergence of solutions. Structured genetic algorithm (SGA) and hierarchical genetic algorithm (HGA) have been proposed to solve optimization problems while simultaneously avoiding the problem of premature convergence (Dasgupta and McGregor 1991-

*Department of Electrical Engineering, National Chung Hsing University, Taichung, Taiwan 402, ROC
E-mail: chunlin@dragon.nchu.edu.tw

1992; Dasgupta and Mcgregor 1992; Tang, Man and Gu 1996; Lai and Chang 2004; Tang, Man, Kwong and Liu 1998). The GAs have also been applied in non-stationary environments (Hlavacek, Kalous and Hakl 2009; Bellas, Becerra and Duro 2009). Hierarchical structure of the chromosomes enables simultaneous optimizations of parameters in different parts of the chromosome structures. Recently, a DNA computing scheme implementing GAs has been developed (Chen, Antipov, Lemieux, Cedeno and Wood 1999).

GAs and SGAs have also been attempted to solve the complicated control design problems (Tang, Man and Gu 1996; Jamshidi, Krohling, Coelho and Fleming 2002; Kaji, Chen and Shibata 2003; Chen and Cheng 1998; Lin and Jan 2005). The two approaches are promising because they waive tedious and complicated mathematical processes by directly resorting to numerical searches for the optimal solution. However, most of the approaches lack a mechanism to optimize the controller's structure (Chen and Cheng 1998) and some approaches (Tang, Man and Gu 1996) require two types of GAs, i.e. BGA and RGA, to simultaneously deal with controller's structure variations and parameter optimization, which is less computationally efficient.

The major emphasis in this chapter is to introduce a variant of traditional GAs, named the real coded structural genetic algorithm (RSGA) (Tsai, Huang and Lin 2009; Liu, Tsai and Lin 2007). It is developed to simultaneously deal with the structure and parameter optimization problems based on a real coded chromosome scheme. For the evolution of generations, it adopts the same crossover and mutation; hence, better computational efficiency results in even evolution of offsprings. For demosntration, two distinguished applications will be introduced, i.e. digital filter and control designs.

For the application of IIR filter design, four types of digital filters are considered, i.e. low pass filter (LPF), high pass filter (HPF), band pass filter (BPF), band stop filter (BSP). The RSGA attempts to minimize specific error within the frequency band of interest and the filter's order to simultaneously ensure performance and structure optimality. Recently, GAs have been attempted to cope with the filter design problems (Tang, Man, Kwong and Liu 1982; Etter, Hicks and Cho 1982). Comparisons show that the results of the RSGA outperform those obtained by the HGA. As to the control application, a robust control design approache for the plant with uncertainties is demonstrated. The RSGA is applied to minimize controller order and consider mixed time/frequency domain specifications, which eases complicated control design methods with practical approaches. To verify the effectiveness of the RSGA, the effect of modeling uncertainty to payload changes is examined. The results obtained from experiments and simulation study prove that the approach is effective in obtaining minimal ordered controllers while ensuring control system performance.

This chapter consists of five sections. In Section 2, introduction of the fundamental GAs such as BGAs and SGAs are given. In Section 3, the RSGA and the operational ideas are explained. Demonstrative applications based on the RSGA are presented in Sections 4 and 5, respectively. Section 6 concludes this chapter.

2. CONVENTIONAL GAs

GAs are widely used to the most feasible solution for many problems in science and engineering. The GAs are artificial genetic systems based on the process of natural selection, and natural genetic. They are a particular class of evolutionary algorithms (also known as evolutionary computation) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination). Unlike traditional search algorithms, GAs don't rely on a large numbers of iterative computational equations to realize the serach algorithms. In the literature, GAs have been widely applied to solve various optimization problems in staionary or non-stationaru environments.

2.1 Binary Genetic Algorithm (BGA)

Fundamental GAs involves three operators: reproduction, crossover, and mutation. Commonly applied GAs are of the binary type. That is, a standard representation of the solution is an array of bits. Regularly, the solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. For BGAs, there are fundamental operations to be implemented as follows.

2.1.1 Coding and Encoding

GAs can be applied for a wide range of problems because the problem evaluation process is completely independent from the problem itself. The only thing needed is a different decoder to interpret the chromosomes in the fitness function. Individuals of BGAs are first encoded into binary strings which consist of 0's and 1's. Precision of each element of k is determined by a string of length l_k and the desired resolution r_k. In general, we have

$$r_k = \frac{U_{\max} - U_{\min}}{2^{l_k} - 1} \tag{1}$$

where $U_{\max}$ and $U_{\min}$ specify the upper and lower limits of the range of parameters, respectively. The decoding process is simply the inverse procedure of the coding process. To avoid spending much time to run the model, researchers favor the use of the base-10 representation because it is needless to encoding or decoding.

2.1.2 Fitness Function

As in the traditional optimization theory, a fitness function in GAs is a particular type of objective function that quantifies the optimality (i.e. extent of "fitness" to the objective function) of a chromosome (solution) so that that particular chromosome may be ranked according to its fitness value against all the other chromosomes. In general, one needs fitness values that are positive; therefore, some kind of monotonically scaling and/or translation may be necessary if the objective function is not strictly positive. Depending on their fitness values, elite chromosomes are allowed to breed and mix other chromosomes by the operation of "crossover" (explained later) to produce a new generation that will hopefully be even "better".

2.1.3 Reproduction

The reproduction operator ensures that, in probability, the better a solution is in the current population, the more replicates it has in next population. Consequently, the reproduction is processed in two stages. In the first stage, the winner-replacing step is performed to raise the possibility for the chance that a chromosome nearby the global optimum is reproduced. In the second stage, a roulette wheel with slots sized proportionally to fitness is spun certain times; each time a single chromosome is selected for a new population.

Reproduction is the process in which the best-fitness chromosomes in the population receive a correspondingly large number of copies according to their fitness value in the next generation, thereby increasing the quality of the chromosomes and, hence, leads to better solutions for the optimization problem. The most common way is to determine selection probability or survival probability for each chromosome proportional to the fitness value, defined as

$$p_i = \frac{f_i^{GA}}{\sum_{k=1}^{n} f_k^{GA}}$$

(2)

The chromosomes that survive to the next generation are placed in a matting pool for crossover and mutation operations.

2.1.4 Crossover

After the parent chromosomes have been selected, the next step is crossover. One can use a crossover operator to generate new chromosomes that retain good features from the previous generation. Crossover is usually applied to select pairs of parents with a probability equal to the crossover rate. Three crossover operators are very common: One-Point, Two-Point, and Uniform Crossover. One-point crossover is the most basic crossover operator which helps select a crossover point on the genetic code at random where two parent chromosomes exchanged at.

Crossover is not usually applied to all pairs of individuals selected for mating. When a random choice is made, it occurs at a certain probability P_C which is referred to as the crossover rate. Usually, a small value for $0 < P_c \leq 1$ is used to ensure that good solutions are not distorted too much. Example of crossover is shown below.

For example, assume that the parent chromosomes *A* and *B* with the crossover point at the fifth bit are given as

↓ **Crossover point**

$A = 0\ 1\ 0\ 1\ 1\ |\ \mathbf{0\ 0\ 0\ 1\ 1}$ **(707)**

$B = 1\ 0\ 0\ 1\ 0\ |\ \mathbf{1\ 0\ 1\ 1\ 0}$ **(1198)**

After crossover, the chromosomes are generated as

↓ **Crossover point**

$A' = 0\ 1\ 0\ 1\ 1\ |\ \mathbf{1\ 0\ 1\ 1\ 0}$ **(750)**

$B' = 1\ 0\ 0\ 1\ 0\ |\ \mathbf{0\ 0\ 0\ 1\ 1}$ **(1155)**

2.1.5 Mutation

Crossover exploits current gene potentials, but if the population does not contain all the encoded information needed to solve a particular problem, then no of gene mixing can produce a satisfactory solution. For this reason, a mutation operator capable of spontaneously generating new chromosomes is needed. The most common way of implementing mutation is to flip one of the bits with a probability. An example of the mutation will be provided later.

The aim of mutation is to introduce new genetic material in an existing chromosome. Mutation also occurs at a certain probability P_m, referred as the mutation rate. Usually, a small value of P_m with $0 < P_m \leq 1$ is used to ensure good solutions are not much distorted. The conventional mutation operator is performed on a gene-by-gene basis. With a given probability of mutation, each gene in all chromosomes from the whole population undergoes mutation.

For example, assume that the chromosome *A* with the mutation point at the sixth bit is described as

$$\downarrow \textbf{Mutation point}$$
$$A = 0\,1\,0\,1\,1\,\mathbf{0}\,0\,0\,0\,1\,1 \quad \textbf{(707)}$$

After mutation, the new chromosome becomes

$$\downarrow \textbf{Mutation point}$$
$$A' = 0\,1\,0\,1\,1\,\mathbf{1}\,0\,0\,0\,1\,1 \quad \textbf{(739)}$$

GAs and other types of variants come in many different varieties, but most are variations and/or elaborations on the following loose outline. GAs typically compute through the following steps, maintaining a population of bitstrings representing candidate solutions of a problem:

- Step 1: Randomly generate N chromosomes on initial population in the search.
- Step 2: Calculate the fitness for each chromosome.
- Step 3: Perform reproduction, i.e. select the better chromosomes with probabilities based on their fitness values.
- Step 4: Perform crossover on chromosomes selected in above step by crossover probability.
- Step 5: Perform mutation on chromosomes generated in above step by mutation probability.
- Step 6: If reach the stop condition or obtain the optimal solution, one may stop the process, else repeat Steps 2-6 until the stop condition is achieved.
- Step 7: Get the optimal solution.

It should be mentioned that this loose outline fits several evolutionary computation paradigms which have varying techniques for fitness evaluation, selection, and breeding.
From the above outline, it can be seen that GAs use the idea of surviving the fittest by passing good genes to the next generation and combine different chromosomes to get new optimal solution.
Other coding types have also been well-discussed in the literature for the representation issue, such as real coded GAs (RGAs), which would seem particularly natural when tackling optimization problems of parameters with variables in continuous or discontinuous domains. In the RGAs, a chromosome is coded as a finite-length string of the real numbers corresponding to the design variables. The RGAs are rigorous, precise, and efficient because the floating point representation is conceptually close to the real design space. In addition, the string length reduces to the number of design variables. Some comparative studies conducted have concluded that the RGAs outperform BGAs in many optimization problems from the aspects of computational precision and speed (Gen and Cheng 2000; Goldberg 1991).

2.2 Structured Genetic Algorithm (SGA)

Conventional SGAs use a binary coding system, only equipped with two functions to represent structure variance: active and inactive (similar to a soft switch). The chromosome structures are coded using real numbers; via structured genetic mapping, the genes on the higher levels determine whether the genes on the lower levels should become activated, deactivated or they should go through linear variation. In SGAs, the primary mechanism to eliminate the dilemma of redundancy is worked through regulatory genes, which switches genes ON (active) and OFF (inactive) respectively.

The central feature of the SGA is its use of redundancy and hierarchy in its genotype. In the structured genetic model, the genomes are embodied in the chromosome and are represented as sets of binary strings. The model also adopts conventional genetic operators and the survival of the fittest criterion to evolve increasingly fit offspring. However, it differs considerably from the simple GAs while encoding information in the chromosome.
The fundamental differences are as follows:

- The SGA interprets chromosomes as hierarchical genomic structures. The two-level structured genomes are illustrated as in Figure 1.
- Genes at any level can either be active or inactive.
- "High level" genes activate or deactivate sets of lower level genes.

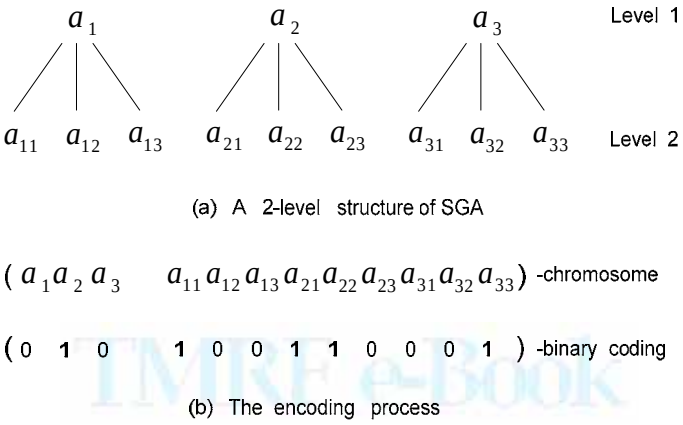


Figure 1. A two-level structure of SGA (Dasgupta and Mcgregor 1992)

Thus, a single change at higher level represents multiple changes at lower levels in terms of genes which are active; it produces an effect on the phenotype that could only be achieved in simple GAs by a sequence of many random changes. Genetic operations altering high-level genes result in changes in the active elements of the genomic structure and hence control the development of the fitness of the phenotype. Basically, SGA woks as a form of long term distributed memory that stores information, particularly genes once highly selected for fitness.
The major advantage of using real-valued coding over binary coding is its high precision in computation. Moreover, the real-valued coding chromosome string is much shorter in the string length.

3. REAL CODED STRUCTURED GENETIC ALGORITHM (RSGA)

To increase computation effectiveness, the RSGA combines the advantages of RGA with SGA. Its chromosome structure is coded using real numbers; via structured genetic mapping, the genes on the higher levels, named control genes, determine whether the genes on the lower levels, named parameter genes, should become activated, deactivated or should the genes go through linear ratio variation between the activated and deactivated statuses.
RSGAs are equipped with three operations: activate, deactivate, and linear ratio variation. The genetic algorithm used in this section is real coded. The real number encoding has been to confirmed to have better performance than binary encoding for constrained optimization. Because of the conventional

SGA uses a binary coding system, only providing two functions to represent structure variance: active and inactive (similar to a soft switch).

3.1 RSGA Operation

Since all chromosomes in the RSGA are real numbers, not only do the control and parameter genes share the same crossover and mutation operations, both genes apply identical crossover and mutation rates during the process of evolution. Therefore, it not only improves computational efficiency, but, most importantly, ensures consistency of the operation of crossover and mutation between control and parameter genes during evolution.

3.2 Structured Genetic Mapping

RSGA consists of two types of genes: control genes and parameter genes, as illustrated in Figure 2. Both of the control genes and parameter genes are real numbers within the range ($R_{\min}, R_{\max}$) where $0.5R_{\max} - R_{\min} > 0$. The activation condition of the control genes is determined by $B_{\min}$ and $B_{\max}$, which corresponds to the boundaries of OFF (inactive) and ON (active) respectively in Figure 3. When the value of the control gene exceeds $B_{\max}$, the corresponding parameter gene is activated; the parameter gene is deactivated when the value of the corresponding control gene is less than $B_{\min}$. When the control gene's value is in between $B_{\max}$ and $B_{\min}$, the corresponding parameter gene is regulated by linearly scaling its original value.

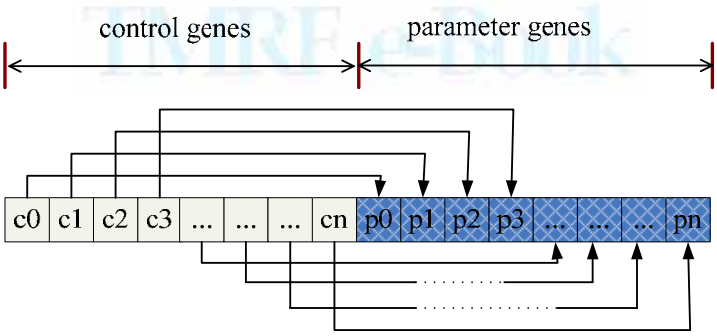


Figure 2. Fundamental chromosome structure of RSGA

For the operational idea depicted above, appropriate values of $B_{\min}$ and $B_{\max}$ have to be determined; however, an improper choice of $B_{\min}$ and $B_{\max}$ may put too much emphasis on structure optimization but neglect emphasizing on parametric optimization, and vice versa. Determination of an effective boundary for $B_{\min}$ or $B_{\max}$ often involves repetitive tests, which is computationally inefficient. In general, one defines $R_{\text{mid}} = 0.5R_{\max} - R_{\min}$, where g_i denotes the current generation, g_{init} and g_{fin} are, respectively, the initial and final generations, and $\Delta B = kg_i > 0$ with k being a positive constant.

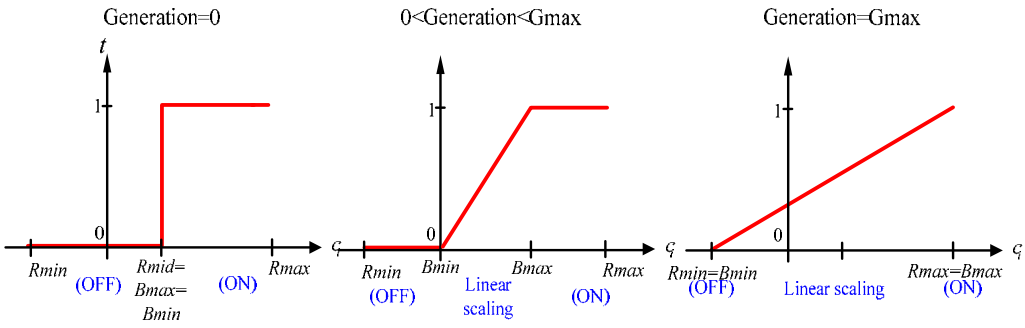


Figure 3. Activation functions of the control genes

The boundary sizing technique can resolve the problem depicted above. The RSGA first acts like a soft switch during the early stage of evolution, provides only ON or OFF functions, emphasizing structural optimization. As the evolution proceeds, the boundary increases progressively with generation number such that linear scaling is put into effect to promote additional emphasis on the parameter optimization. At the end of the generation, the boundaries of $B_{\min}$ and $B_{\max}$ will replace the entire range, focusing only on parameter optimization. This ensures that this structure is optimized before shifting the focus into the optimization of parameters.

To explain, the mathematical model of the fundamental chromosome structure is defined as follows

$$X =< C, P >= ([c_i], [p_i]),\ c_i \in \square,\ p_i \in \square^{1\times n_{pi}},\ i = 1,\ldots,n \tag{3}$$

where $X = [x_i]_{1\times\left(n+\sum_{i=1}^{n} n_{pi}\right)}$ represents an ordered set consisting of the control genes' strings $C = [c_i]_{1\times n}$ and the parameter genes' string $P = [p_i]_{1\times\left(\sum_{i=1}^{n} n_{pi}\right)}$. The structured genetic mapping from C to P is defined as

$$\tilde{X} =< C, \tilde{P} >=< C, C \otimes P > \tag{4}$$

where $\tilde{P} = [c_i] \otimes [p_i]$ with

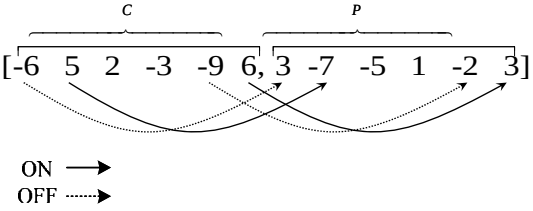
$$\tilde{p}_i \square \begin{cases} p_i, & \text{if } B_{\max} \le c_i \\ p_i t, & \text{if } B_{\min} < c_i < B_{\max} \\ \phi, & \text{if } c_i \le B_{\min} \end{cases} \tag{5}$$

$\otimes$ is called the soft genetic switch which determines the status of each element in the parameter gene string (determined by (5)), ϕ is the bit denoting the corresponding parameter gene to be ignored in the later evolution process, $t = \dfrac{c_i - B_{\min}}{B_{\max} - B_{\min}}$ denotes the effective gain for c_i within the boundary of $B_{\min}$ and $B_{\max}$.

Example:

A randomly generated chromosome is given in the follows where $B_{\max} = 5$ and $B_{\min} = -5$.

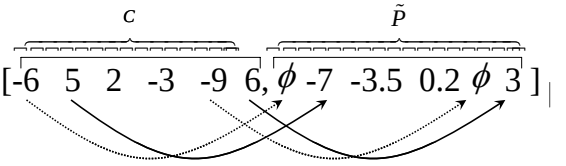
$X = < C, P >$



Since

$$c_1 = -6 < B_{\min} = -5 \Rightarrow \text{OFF} \rightarrow \tilde{p}_1 = \phi$$
$$c_2 = 5 \geq B_{\max} = 5 \Rightarrow \text{ON} \rightarrow \tilde{p}_2 = p_2 = -7$$
$$c_3 = 2,\ B_{\min} < 2 < B_{\max} \rightarrow \tilde{p}_3 = p_3 t_3 = -3.5, \quad \text{where } t_3 = 0.7$$
$$c_4 = -3,\ B_{\min} < -3 < B_{\max} \rightarrow \tilde{p}_4 = p_4 t_4 = 0.2, \quad \text{where } t_4 = 0.2$$
$$c_5 = -9 < B_{\min} = -5 \Rightarrow \text{OFF} \rightarrow \tilde{p}_5 = \phi$$
$$c_6 = 6 > B_{\max} = 5 \Rightarrow \text{ON} \rightarrow \tilde{p}_6 = p_6 = 3$$

$X = < C, P > \rightarrow\rightarrow \tilde{X}$:



3.3 Genetic Mathematical Operations

Similar to standard GAs, the RSGA involves three major genetic operations: selection, crossover, and mutation. Since both control genes and parameter genes are real numbers, unlike the conventional SGAs (Tang, Man and Gu 1996), the RSGA utilizes the same mathematical operations to conduct crossover and mutation, which is less time consuming and requires less computational burden. In conventional SGAs, the control and parameter genes were coded as binary and real numbers, respectively; therefore, the crossover and mutation for the two parts require different mathematical mechanism which may result in uneven evolution.

3.3.1 Initial Population

As for RGSs, the initial population consisting a certain amount of randomly generated individuals (in the form of (3)), each represented by a real numbered string, is created. Each of these strings (or chromosomes) represents a possible solution to the search problem.

3.3.2 Fitness

The algorithm works toward optimizing the objective function, which consists of two parts corresponding, respectively, to structure and parameters as follows:

$$J_{tot}(X) = \rho J_s(X) + (1-\rho) J_p(X) \tag{6}$$

with J_s being the normalized index of structure complexity; J_p, the normalized performance index; $\rho \in [0,1]$ being the weighting factor representing the desired emphasis on structure complexity.

To employ the searching algorithms, the targeted solutions are commonly related to the fitness function $f(\cdot)$. A linear equation can be introduced to relate the total cost index $J_{tot}(\cdot)$ to $f(\cdot)$. The linear mapping between $J_{tot}(\cdot)$ and $f(\cdot)$, called as the "windowing" technique, is given as

$$f(\cdot) = \mu \left[J_{tot}(\cdot) - J_l \right] + f_u \tag{7}$$

which $\mu = \dfrac{f_u - f_l}{J_l - J_u}$, with J_u and J_l denoting the largest and smallest values of $J_{tot}(\cdot)$ within all individuals evaluated in the current generation, and f_l and f_u being the constant minimum and maximum fitness values assigned, respectively, to the worst and best chromosomes.

3.3.3 Selection

In this phase, a new population is created from the current generation. The selection operation determines which parent participates in the production of offsprings for the next generation, analogous to the survival of the fittest in the process of natural selection. Usually, members selected for mating depends on their individual fitness values. A common way to implement selection is to set the selection probability equal to $f_i \Big/ \sum_{j=1}^{m} f_j$, where f_i is the fitness value of the *i*-th member, and *m* is the population size. A higher probability tends to be assigned to chromosomes associated with higher fitness value among the population.

3.3.4 Crossover

Crossover plays the role of mixing two chosen chromosomes as in RGAs. As in the usual GAs, the number of individuals joining the operation is determined by the crossover probability p_c, usually $0.5 \le p_c \le 0.9$. The crossover of the randomly selected pairs of individuals is a combination of an extrapolation/interpolation method with a crossover process. It starts from extrapolation and shifts to interpolation if the parameters of the offspring exceed the permissible range. Interpolation avoids parameters from going over the admissible range during the boundary value search. The operation is determined by

$$\begin{cases} \tilde{x}_{di} = x_{di} - \lambda \left(x_{di} - x_{mi} \right), \\ \tilde{x}_{mi} = x_{mi} + \lambda \left(x_{di} - x_{mi} \right), \end{cases} \text{ if } x_{di} > R_{\max} \text{ or } x_{mi} < R_{\min}, \tag{8}$$

$$\begin{cases} \tilde{x}_{di} = x_{di} + \lambda\left(x_{di} - x_{mi}\right), \\ \tilde{x}_{mi} = x_{mi} - \lambda\left(x_{di} - x_{mi}\right), \end{cases} \text{ if } R_{\min} \le x_{mi} \le x_{di} \le R_{\max}, \tag{9}$$

$$x_{di}, x_{mi} \notin \{\phi\},\ j \le i \le 2n,\ 1 < j < 2n \tag{10}$$

$$\lambda = \lambda_0\left(1 - \frac{g_i}{g_{fin}}\right),\ x_{di} \ge x_{mi} \tag{11}$$

where the subscripts m and d are used to discriminate the parents: mom and dad, i denotes the *i*-th elements in the parent chromosomes, ($\tilde{x}_d$, $\tilde{x}_m$) are offsprings of (x_d, x_m), $\lambda_0 \in [0,1]$ is a uniform random number, the parameter j is a random number in the pair of parents to designate the crossover point:

$$j = \text{ceil}(2n \times \lambda_0) \tag{12}$$

Note that both control and parameter genes apply the same crossover operation to generate new individuals. This unified mechanism avoids the need to utilize two types of crossover operators with different attributes as required in traditional SGAs. The unified mechanism results in better computation efficiency and consistency of chromosome crossover.

3.3.5 Mutation

The mutation operator applies individually for the randomly chosen individuals. The number of individuals to be varied is determined by the mutation probability p_m, $0 < p_m \le 1$. The operator adopts the technique of non-uniform mutation which changes the gene in a chromosome with each generation via

$$\tilde{x}_{ij} = \begin{cases} x_{ij} + \Delta(g, x_{ij\max} - x_{ij}), & \text{if } h = 0 \\ x_{ij} - \Delta(g, x_{ij} - x_{ij\min}), & \text{if } h = 1 \end{cases} \tag{13}$$

$$\Delta(g_i, y) = \beta\lambda_0\left(1 - \frac{g_i}{g_{fin}}\right) \tag{14}$$

where $x_{ij} \notin \{\phi\}$, the random integer number $h \in \{0,1\}$, $x_{ij\max}$ ($x_{ij\min}$) is the maximal (minimal) *j*-th element of the *i*-th individual in the current generation, and β is a scaling factor.

The above mechanism is employed so that the control genes affect the parameter genes during the early stage of evolution and optimize the structure. As the process of evolution progresses, the control genes loose their effect; the algorithm gradually tends to optimize the parameter genes.

The structured genetic mapping for all individuals, from the control genes to parameter genes, defined by (4) is performed after mutation. A new generation with m offsprings is bred. The process repeats until the fittest individual is generated.

3.3.6 Termination Criterion

The algorithm produces new generations until the terminal condition is reached. The algorithm terminates when the generation number reaches a specific number or the fitness value of the best fitted individual converge to an acceptable level. And then the individual with maximum fitness function is chosen to be the solution.

4. FILTER DESIGN USING RSGA

This section demonstrates a useful application of RSGAs in filter designs, in which the objective is incorporated to the algorithm to tackle the digital IIR filter design.

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise, or extract useful parts of the signal, such as the components lying within a certain frequency range. A typical digital filter design problem involves determination of a set of filter coefficients to meet certain specifications such as the width of the passband (ω_p) and the corresponding gain, the width of the stopband (ω_s) and the attenuation therein, the band edge frequencies and the peak ripple tolerable magnitudes (δ_p , δ_s) in the passband and stopband. Figure 4 shows related filtering specifications for the low pass filter (LPF), high pass filter (HPF), band pass filter (BPF), and band stop filter (BSP).

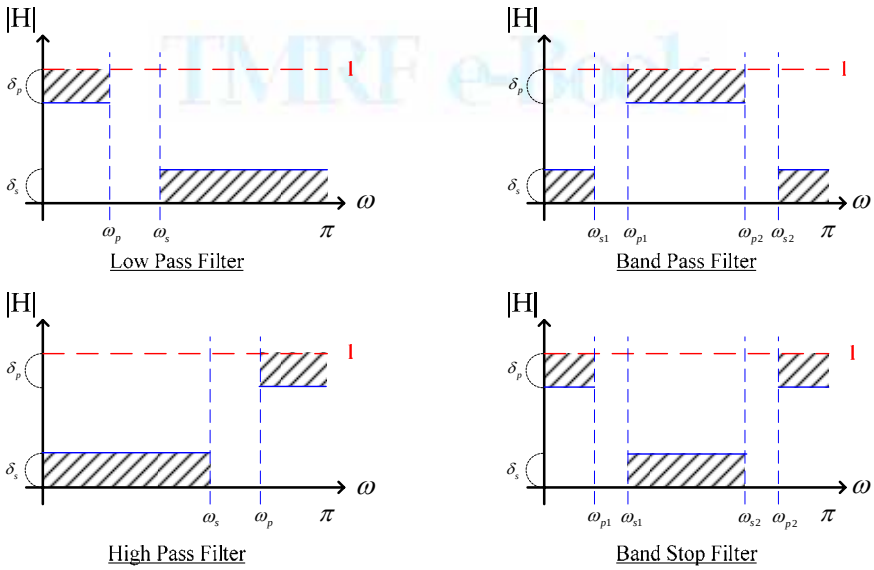


Figure 4. Specifications of LP, HP, BP, and BS filters

Two types of digital filters are commonly encountered, i.e. infinite impulse response (IIR) and finite impulse response (FIR) filters. An IIR filter has an impulse response function that is non-zero over an infinite length of time. This is in contrast to the FIR filter, which possesses fixed-duration impulse responses. An IIR filter may continue to respond indefinitely, whereas the impulse response of an Nth-order FIR filter only lasts for N+1 sampling periods.

4.1 IIR Filtering Model (Shynk 1989)

In the time domain, the generic N th-order difference equation defines how the output signal is related to the input signal that represents an IIR filter described by

$$y[n]-\sum_{i=1}^{N} a_i y[n-i]=\sum_{i=0}^{M} b_i x[n-i] \tag{15}$$

where M is the feedforward filter order, N is the feedback filter order, b_i are the feedforward filter coefficients, a_i are the feedback filter coefficients. This equation shows how to compute the next output sample $y[n]$, in terms of the past outputs $y[n-i]$, the present input $x[n]$, and the past inputs $x[n-i]$. Applying the filter to an input in realization form is depending on the exact order of evaluation.

The transfer function (TF) $H(z)$ of the IIR filter possesses the following form:

$$H(z)=\frac{B(z)}{A(z)}=\frac{b_0+b_1z^{-1}+b_2z^{-2}+\cdots+b_Mz^{-M}}{1+a_1z^{-1}+a_2z^{-2}+\cdots+a_Nz^{-N}},\ M\le N \tag{16}$$

or

$$H(z)=K\frac{\prod_{i=1}^{M_1}(z+b_j)\prod_{i=1}^{M_2}(z^2+b_{l1}z+b_{l2})}{\prod_{i=1}^{N_1}(z+a_i)\prod_{i=1}^{N_2}(z^2+a_{k1}z+a_{k2})},\ M\le N \tag{17}$$

where the orders of the numerator and denominator polynomials in (17), according to the degree of z, are, respectively, $M=M_1+2M_2$ and $N=N_1+2N_2$, K is a constant, and a_i, b_i, a_{i1}, a_{i2}, b_{i1}, b_{i2} are coefficients of the corresponding terms.

4.2 Constraints on IIR Digital Filter

- The order of the denominator polynomial is larger than or equal to that of the numerator polynomial:

$$\deg(A(z))\ge\deg(B(z)) \tag{18}$$

where $\deg(p(z))$ denotes the degree of the polynomial $p(z)$.

- To ensure stability, the roots of the denominator polynomial (i.e. poles of $H(z)$) must lies inside the unit circle:

$$\text{mag}\left[\text{poles}(H(z))\right] \le 1 \tag{19}$$

where $\text{poles}(H(z))$ denotes the poles of $H(z)$.

4.3 Digital IIR Filter Design

A typical digital filter design problem involves the determination of a set of filter coefficients which meet performance specifications, such as passband width and corresponding gain, widths of the stopband and attenuation, band edge frequencies, and tolerable peak ripple in the passband and stopband. The RSGA is capable of coding the system parameters into a hierarchical structure.
A complete gene string representing $H(z)$ consists of $M_1+N_1+M_2+N_2+1$ control genes and $M_1+N_1+2(M_2+N_2)+1$ parameter genes. The coefficients of the first- and second-order models are controlled by control genes, which determine the corresponding coefficients whether to turn ON or OFF or should they go through linear scaling.

The performance index J_{tot} in (6) for an optimal digital IIR filter design is defined as

$$J_{tot}(\cdot) = \rho J_s(\cdot) + (1-\rho)\left[J_{pp}(\cdot) + J_{ps}(\cdot)\right] \tag{20}$$

where

$$J_s = \frac{1}{\delta}\text{Log}(N_{\max} - N + 1) \tag{21}$$

with $\delta = \max\{\varepsilon, \text{sgn}(N-M)\}$, ε, a small positive constant, $N_{d\max}$, the assigned maximum order of the denominator polynomial; J_{pp} and J_{ps} ensure that the frequency responses are confined within the prescribed frequencies listed in Table 1; for each filter specification, the tolerable error of passband and stopband are defined as

$$J_{pp} = \begin{cases} \sum_{pb=1}^{r_p} \left| \left| H(e^{j\omega_{pb}}) \right| - 1 \right|, & \text{if } \left| H(e^{j\omega_{pb}}) \right| > 1 \\ \sum_{pb=1}^{r_p} \left| 1-\delta_p - \left| H(e^{j\omega_{pb}}) \right| \right|, & \text{if } \left| H(e^{j\omega_{pb}}) \right| < 1-\delta_p \end{cases}, \ \forall \omega_{pb} \in \text{passband} \tag{22}$$

and

$$J_{ps} = \sum_{sb=1}^{r_s} \left| \left| H(e^{j\omega_{sb}}) \right| - \delta_s \right|, \ \text{if } \left| H(e^{j\omega_{sb}}) \right| > \delta_s, \ \forall \omega_{sb} \in \text{stopband} \tag{23}$$

with δ_p and δ_s representing the ranges of the tolerable error for bandpass and bandstop, respectively; r_p and r_s are the numbers of equally spaced grid points for bandpass and bandstop, respectively. The magnitude response is represented by discrete frequency points in the passband and stopband.

4.4 Simulation Study

The first experiment is to design digital filters that maximize the fitness function within certain frequency bands as listed in Table 1.

The IIR filter models originally obtained are as follows (Tang, Man, Kwong and Liu 1982)

$$H_{LP}(z) = 0.0386\frac{(z+0.6884)(z^2-0.0380z+0.8732)}{(z-0.6580)(z^2-1.3628z+0.7122)}$$

$$H_{HP}(z) = 0.1807\frac{(z-0.4767)(z^2+0.9036z+0.9136)}{(z+0.3963)(z^2+1.1948z+0.6411)}$$

$$H_{BP}(z) = 0.077\frac{(z-0.8846)(z+0.9033)(z^2-0.0498z-0.8964)(z^2+0.031z-0.9788)}{(z+0.0497)(z-0.0592)(z^2-0.5505z+0.5371)(z^2+0.5551z+0.5399)}$$

$$H_{BS}(z) = 0.4919\frac{(z^2+0.4230z+0.9915)(z^2-0.4412z+0.9953)}{(z^2+0.5771z+0.4872)(z^2-0.5897z+0.4838)}$$

For comparison, the RSGA was applied to deal with the same filter design problem. The fitness function defined in (20) was adopted to determine the lowest ordered filter with the least tolerable error. The weighting factor ρ was set to 0.5 for both LPF and HPF, and 0.7 for BPF and BSF. M and N were set to be 5. The genetic operational parameters adopted the same settings as those of the HGA proposed by Tang *et al.* (1982). The population size was 30, the crossover and mutation probabilities were 0.8 and 0.1, respectively, and the maximum iteration were 3000, 3000, 8000, and 8000 for LPF, HPF, BPF and BSF, respectively. By running the design process 20 times, the best results using the RSGA approach were

$$H_{LP}(z) = 0.054399\frac{(z+0.2562)(z^2-0.448z+1.041)}{(z-0.685)(z^2-1.403z+0.7523)}$$

$$H_{HP}(z) = 0.063462\frac{(z-1.341)(z^2+0.8952z+0.9913)}{(z+0.5766)(z^2+1.332z+0.7277)}$$

$$H_{BP}(z) = 0.071458\frac{(z+1.362)(z+1.048)(z-1.132)(z-0.6244)}{(z^2-0.4969z+0.6853)(z^2+0.4894z+0.6929)}$$

$$H_{BS}(z) = 0.44428\frac{(z^2 + 0.3241z + 0.8278)(z^2 - 0.3092z + 0.8506)}{(z^2 - 0.8647z + 0.5233)(z^2 + 0.8795z + 0.5501)}$$

For further comparison, define ε_p and ε_s as the maximal ripple magnitudes of passband and stopband:

$$\varepsilon_p = 1 - \min\left\{\left|H(e^{j\omega_{pb}})\right|\right\},\ \forall\omega_{pb} \in \text{passband} \tag{24}$$

$$\varepsilon_s = \max\left\{\left|H(e^{j\omega_{sb}})\right|\right\},\ \forall\omega_{sb} \in \text{stopband} \tag{25}$$

Table 1. Prescribed frequency bands

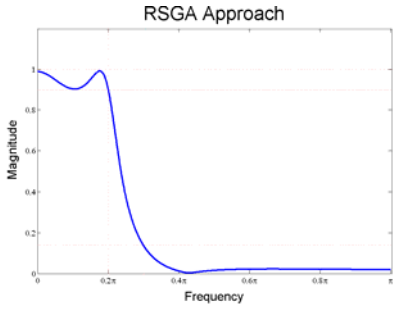
Filter Type	Bandpass	Bandpstop
LP	$0 \le \omega_{pb} \le 0.2\pi$	$0.3\pi \le \omega_{sb} \le \pi$
HP	$0.8\pi \le \omega_{pb} \le \pi$	$0 \le \omega_{sb} \le 0.7\pi$
BP	$0.4\pi \le \omega_{pb} \le 0.6\pi$	$0 \le \omega_{sb} \le 0.25\pi$ $0.75\pi \le \omega_{sb} \le \pi$
BS	$0 \le \omega_{pb} \le 0.25\pi$ $0.75\pi \le \omega_{pb} \le \pi$	$0.4\pi \le \omega_{sb} \le 0.6\pi$

The filtering performances within the tolerable passband and stopband are summarized in Table 2. The ranges of the tolerable error for bandpass δ_p and bandstop δ_s in the RSGA approach are less than the HGA. Figure 5 shows the frequency responses of LPF, HPF, BPF and BSF with two approaches. The RSGA yielded filters with lower passband ε_p and stopband ε_s ripple magnitudes, see Table 3.

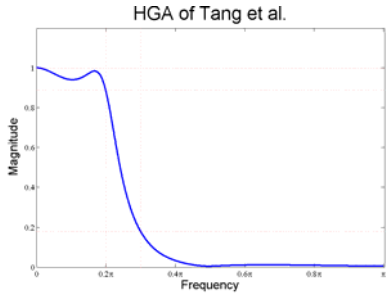
Table 2. Comparison of filtering performance

RSGA Approach	
Passband Tolerance	Stopband Tolerance
$0.9 \le \left|H(e^{j\omega_{pb}})\right| \le 1$ $(\delta_p = 0.1)$	$\left|H(e^{j\omega_{sb}})\right| \le 0.15$ $(\delta_s = 0.15)$

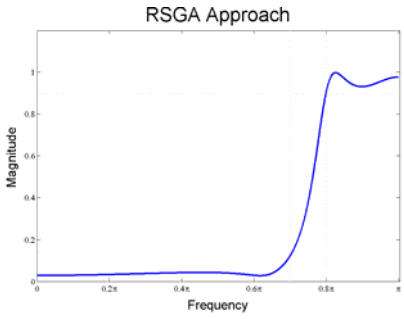
HGA Approach	
Passband Tolerance	Stopband Tolerance
$0.89125 \le \left\|H(e^{j\omega_{pb}})\right\| \le 1$ $(\delta_p = 0.10875)$	$\left\|H(e^{j\omega_{sb}})\right\| \le 0.17783$ $(\delta_s = 0.17783)$



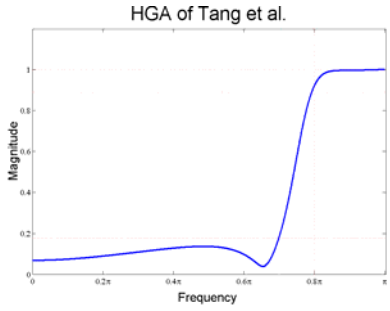
(1a)



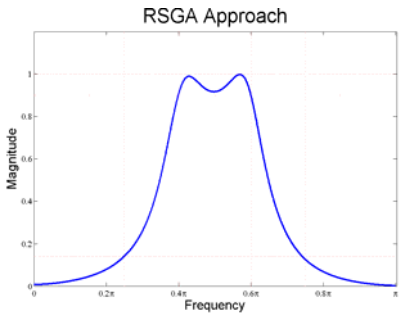
(1b)



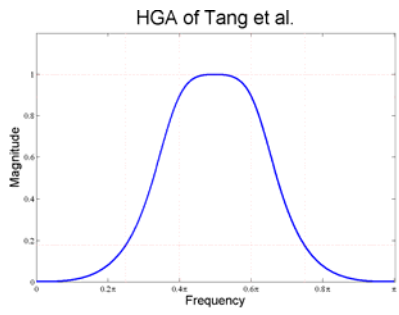
(2a)



(2b)



(3a)



(3b)

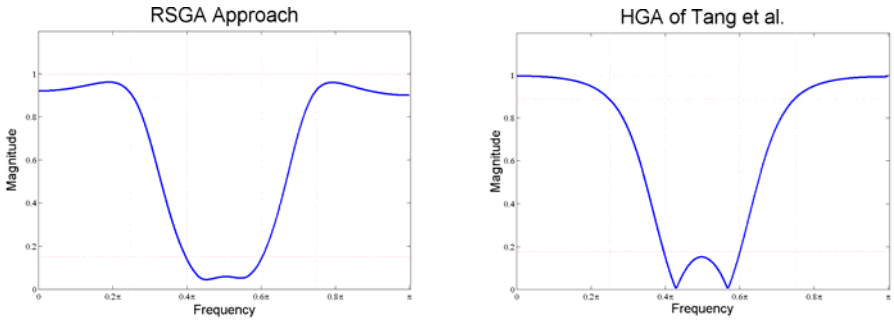


Figure 5. Frequency responses of LPF (1a, 1b), HPF (2a, 2b), BPF (3a, 3b), BSF (4a, 4b) by using the RSGA and HGA approaches

Table 3. Results for LPF, HPF, BPF, BSF

RSGA Approach		
Filter Type	Passband Ripple Magnitude ε_p	Stopband Ripple Magnitude ε_s
LP	0.0960	0.1387
HP	0.0736	0.1228
BP	0.0990	0.1337
BS	0.0989	0.1402
HGA Approach		
Filter Type	Passband Ripple Magnitude ε_p	Stopband Ripple Magnitude ε_s
LP	0.1139	0.1802
HP	0.0779	0.1819
BP	0.1044	0.1772
BS	0.1080	0.1726

Table 4 shows that the RSGA approach requires less generation to achieve the desired design specifications under the same design requirements. Table 5 shows that the resulting BP filter is structurally simpler.

Table 4. Required generation number for convergence

Filter Type	Generation	
	RSGA Approach	HGA Approach
LP	1242	1649
HP	879	1105
BP	3042	3698
BS	4194	7087

Table 5. Lowest Filter Order

Filter Type	Filter order	
	RSGA Approach	HGA Approach
LP	3	3
HP	3	3
BP	4	6
BS	4	4

5. CONTROL DESIGN USING RSGA

This section presents one of the major applications of RSGAs in control design.

5.1 Control Design Description

Consider the closed-loop control system as shown in Figure 6, where $P_n(s)$ is the nominal plant model; $\Delta P(s)$, the multiplicative unmodeled dynamics; $C(s)$, the controller; $H(s)$, the sensor model; $d(t)$, the reference input; $u(t)$, the control command; $y(t)$, the plant output; $e(t)$, the tracking error. Modeling inaccuracy is considered due to uncertain effects or modeling errors.

Let the uncertain plant model $P(s)$ be described by

$$P(s) = P_n(s)\left[1+\Delta P(s)\right] \tag{26}$$

where $\Delta P(s)$ satisfies

$$\left|\Delta P(j\omega)\right| \le \left|W_t(j\omega)\right|, \ \forall\omega \tag{27}$$

In which is a stable rational function used to envelop all possible unmodeled uncertainties.

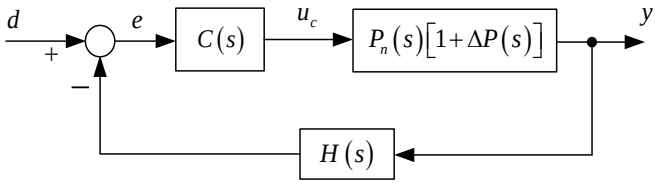


Figure 6. Typical uncertain closed-loop feedback control scheme

The chromosome structure of RSGA representation in Figure 2 is applied to deal with the robust control design problem where the generalized TF of a robust controller is given by

$$C(s) = K \frac{\prod_{j=1}^{\tilde{M}_1}(s+b_j)\prod_{l=1}^{\tilde{M}_2}(s^2+b_{l1}s+b_{l2})}{\prod_{i=1}^{\tilde{N}_1}(s+a_i)\prod_{K=1}^{\tilde{N}_2}(s^2+a_{k1}s+a_{k2})} \tag{28}$$

where K is a constant, a_i , b_j , a_{k1} , a_{k2} , b_{l1} , b_{l2} are the coefficients of the corresponding polynomials.

5.1.1 Frequency Domain Specifications

H_∞ control theory is a loop shaping method of the closed-loop system that H_∞-norm is minimized under unknown disturbances and plant uncertainties (Green and Limebeer 1995). The principal criterion of the problem is to simultaneously consider robust stability, disturbance rejection as well as control consumption.

5.1.1.1 Sensitivity Constraint

The problem can be defined as

$$G_1 \triangleq \left\| W_s S(j\omega) \right\|_\infty < 1 \tag{29}$$

where $\|A(j\omega)\|_\infty = \sup_\omega |A(j\omega)|$ with $A(s)$ being a stable rational function; $S(s) = 1/(1+P_n(s)C(s)H(s))$, the sensitivity function; $W_s(s)$, the weighting function that possesses a sufficiently high gain in low frequency to achieve disturbance suppression and eliminate the steady-state error.

5.1.1.2 Control Constraint

To fulfill the requirement of minimizing control commands at higher frequency, an appropriate weighting function $W_r(s)$ should be introduced. To fulfill the constraint, the following criterion is considered:

$$G_2 \triangleq \left\| W_r R(j\omega) \right\|_\infty < 1 \tag{30}$$

where $R(s) = C(s)/(1+P_n(s)C(s)H(s))$ is the control TF for the nominal closed-loop system. The inequality guarantees internal stability if there is no right-half plane pole/zero cancellation in the controller and plant.

5.1.1.3 Robust Stability Constraint

The controller $C(s)$ is required to maintain closed-loop stability when encountering uncertainty $\Delta P(s)$; the requirement can be assured when the following inequality is satisfied:

$$\left|\Delta P(j\omega)\right| < \frac{1}{\left|T(j\omega)\right|},\ \forall\omega \tag{31}$$

where $T(s) = 1 - S(s)$ is the complementary sensitivity function.

Under the assumption (31), the robust stability constraint can be expressed as

$$G_3 \square \left\|W_t T(j\omega)\right\|_\infty < 1 \tag{32}$$

From the viewpoint of the weighting function, $W_t(s)$ defined in (27) works such that the resulting controller does not stimulate higher resonant frequency of the plant $P(s)$.

5.1.2 Weighting Functions

In a nominal closed-loop system, if the bandwidth is required to be at ω_c then the requirement can be assured by introducing an appropriate weighting function $W_t(s)$ with high gain at higher frequencies and the corner frequency close to ω_c.

In addition to consider disturbance rejection, $W_s(s)$ is designed to eliminate the steady-state error. To make the steady-state error equal to zero with respect to the step input $d(t)$, one must have $\left|S(0)\right| = 0$, i.e. the weight should approach infinity when ω approaches zero. Under the invariant relationship $S(j\omega) + T(j\omega) = 1$, $\forall\omega$, the corner frequency ω_c can be approximately chosen to complement the asymptotic frequency property of $W_t(s)$. The design of the weighting function $W_r(s)$ is similar to that of $W_t(s)$, where the objective is to avoid excessive control commands at higher frequencies.

5.1.3 Time Domain Specifications

Evaluation of the transient response can be performed via an effective method by relating directly to the tracking performance, opposed to the conventional method of examining frequency domain specifications indirectly.

The objective function can be chosen to reflect the overall performance

$$\tilde{J}_p = \alpha_1 M_O + \alpha_2 Tr + \alpha_3 Ess,\ \sum_{i=1}^{3}\alpha_i = 1 \tag{33}$$

where $\alpha_i \geq 0$ are the scalars included to weigh the relative importance of three transient response indices; M_O is the normalized maximum overshoot, Tr is the normalized rise time, and Ess is the normalized steady-state error.

5.1.4 Multiple-Objective Optimization

Multiple-objective optimization problems are usually quite different from the single-objective ones. One attempts to obtain the best solution which is absolutely superior to all other alternatives. This section considers a performance index to reflect the control requirement and overall system

performance, i.e. sensitivity suppression, control consumption, good transient response, and simpler controller's structure. The objective is to search for an appropriate controller that makes each objective function achieve the best compromise.

Although a complicated H_∞ controller usually possesses superior performance, a simpler structure with acceptable performance is more practical. Therefore, the objective functions considered should contain not only performance but also structural information of the controller. For the current problem, define a generalized constrained multiobjective model as follows

$$\min \tilde{F}_O(\cdot) \tag{34}$$

s.t.

$$G_i(\cdot) \le 1,\ i = 1, 2, 3 \tag{35}$$

and the multiobjective function are defined as follows

$$\tilde{F}_O = \beta \tilde{J}_s + (1-\beta)\tilde{J}_p \tag{36}$$

where

$$\tilde{J}_s = \frac{\tilde{n}_d}{\tilde{N}_m},\ \beta = \beta_0 \left(e^{-q\frac{g}{g_{\max}}} + \tilde{\sigma} \right)$$

$\tilde{J}_p$ is time domain response measure, $\tilde{J}_s$ is the normalized controller order, $\tilde{n}_d$ is the order of the denominator polynomial, $\beta_0 \in [0,1]$ is a weighting factor that represents the desired emphasis on structure complexity, g is the current generation, $g_{\max}$ is the maximum number of generations, q is a shaping factor, and $\tilde{\sigma}$ is a small positive constant which avoids the weight of $\tilde{J}_s$ from zero.

The minimization formulation can be transformed into a fitness evaluation $\tilde{f}(\cdot)$ which is a measure evaluating suitability of a candidate and is to be used in the solution searching process. The fitness value determines the probability that a candidate solution is selected to produce possible solutions in the next generation.

To apply the RSGA, a linear equation is introduced to connect $\tilde{F}_O(\cdot)$ and $\tilde{f}(\cdot)$ as follows

$$\tilde{f}(\cdot) = \mu \tilde{F}_O(\cdot) + \zeta \tag{37}$$

where $\mu = \dfrac{\tilde{f}_u - \tilde{f}_l}{\tilde{F}_{Ol} - \tilde{F}_{Ou}}$ and $\zeta = \tilde{f}_u - \mu$ with $\tilde{F}_{Ou}$ and $\tilde{F}_{Ol}$ being the largest and smallest values evaluated in the current generation, and $\tilde{f}_u$ and $\tilde{f}_l$ being the corresponding fitness values, respectively.

Then, applying the penalty function makes infeasible solutions have less fitness than the feasible ones. To achieve this aim, a constrained multi-objective fitness function can be applied:

$$\tilde{F}(\cdot) = \tilde{f}(\cdot)\, p(\cdot) \tag{38}$$

where the penalty function $p(\cdot)$ is applied for adjusting the value of penalty in the constrained multiple-objective function at each generation and to differentiate feasible and infeasible candidates:

$$p(\cdot) = 1 - \frac{1}{3}\sum_{i=1}^{3}\left(\frac{\Delta C_i(\cdot)}{\Delta C_i^{\max}(\cdot)}\right) \tag{39}$$

and

$$\Delta C_i(\cdot) = \max\{0, G_i(\cdot) - 1\}, \quad \Delta C_i^{\max}(\cdot) = \max\{\eta_p, \Delta C_i(\cdot)\}$$

in which $\Delta C_i(\cdot)$ is the value of violation for *i-th* constraint, $\Delta C_i^{\max}$ is the maximum violation for *i*-th constraint in the current population, and η_p is a small positive number. One applies the penalty function to adjust the values of penalties in the constrained multi-objective fitness function at each generation to differentiate between feasible and infeasible parameters. The method raises efficiency in the searching for optimal solution.

5.2 Simulation Study

5.2.1 Robust Control for Non-Minimum Phase Plant

Consider a typical H_∞ control design problem with control constraints. The non-minimum phase plant with a right-half plane zero is given by

$$P(s) = \frac{s-2}{(s+0.01)(s-3)}$$

and the weighting function corresponding to the control constraint is

$$W_r(s) = \frac{0.01s + 0.1}{s + 100}$$

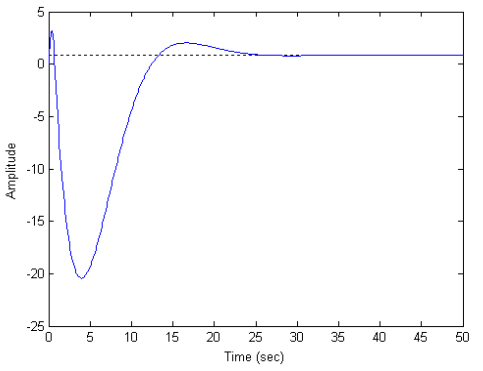
Synthesis of the H_∞ control problem subject to the constraint (30) was solved by applying Matlab-Robust Control Toolbox (Balas, Chiang, Packard and Safonov 1995) and the RSGA. The former yielded an optimal H_∞ controller given as

$$C_{Matlab}(s) = \frac{2.486\cdot 10^4 s^2 + 2.486\cdot 10^6 s - 2488}{s^3 + 1.094 s^2 + 1.5\cdot 10^5 s - 1.659\cdot 10^6}$$

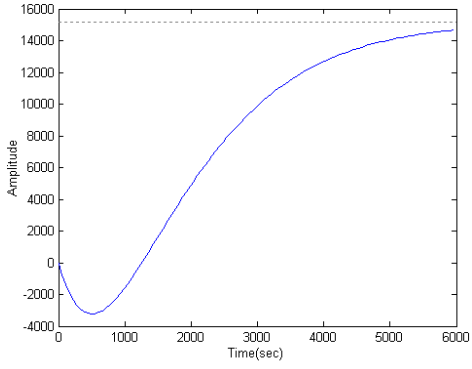
For the RSGA approach, the population size was chosen to be 30; crossover rate, $P_C = 0.8$; mutation rate, $P_m = 0.2$; the maximum number of generations, 100; $\rho_0 = 0.5$. The resulting controller was obtained as

$$C_{RSGA}(s) = \frac{230.2555(s - 0.006682)}{s^2 + 12.057s - 159.139}$$

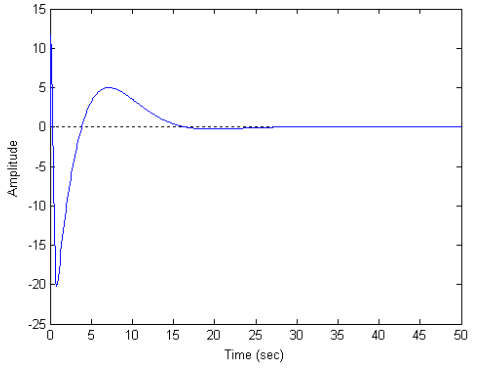
This gives G_2 in (30) as 0.0348. The results of transient responses displayed in Figure 7 show better performance of the RSGA approach than that obtained by Matlab-Robust Control Toolbox. Figure 7(1) shows faster response, Figure 7(2) show smaller control commands, and Figure 7(3) shows less tracking errors of the RSGA design.



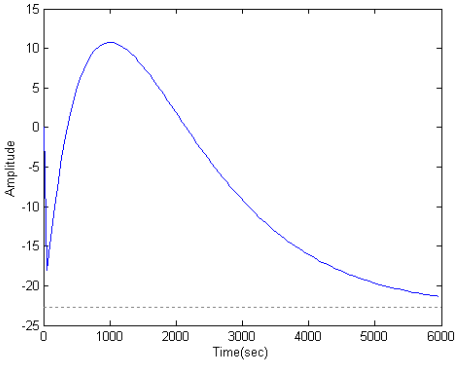
(1a)



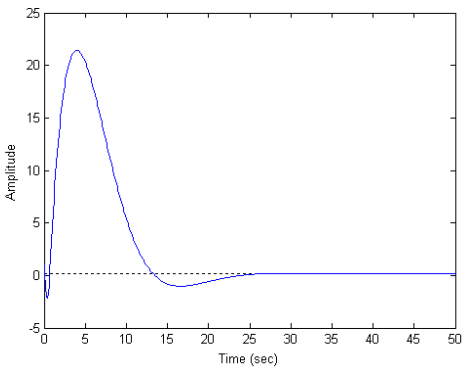
(1b)



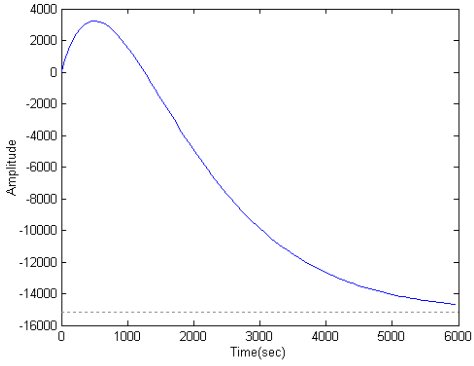
(2a)



(2b)



(3a)



(3b)

Figure 7. Step response (1a, 1b), control command (2a, 2b) and tracking error (3a, 3b) for the unstable plant with a non-minimum phase zero by using the RSGA approach and MATLAB-Robust Control Toolbox

5.2.2 Control Design for MBDCM

The TF of a micro brushless DC motor (MBDCM) Pittman Express 3442S004-R3 for precision positioning, while no load, is given by

$$P(s) = \frac{0.0486}{7.47\times10^{-8}s^3 + 0.0006s^2 + 0.1097s}$$

The weighting functions of the sensitivity function and the control TFs were chosen, respectively, as follows

$$W_s(s) = \frac{100}{500s+1},\ W_r(s) = \frac{0.25}{s^2+10s+100}$$

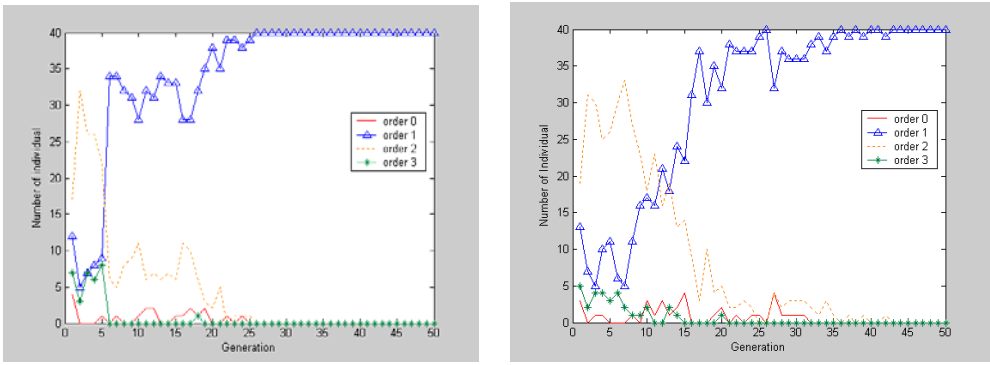
Figure 8 compares the structural variation between the RSGA and SGA during their evolution processes over generations. As it was displayed, the former exhibits more structural variations in the earlier generations but switched to optimizing the parameters later on. The conventional SGA also exhibits adequate structural variations; however, the structure converges slower than that of the RSGA. The controllers obtained by using the RSGA and SGA were given, respectively, as follows

$$C_{RSGA}(s) = \frac{31.8895(s+0.03835)}{s+0.03213},\ C_{SGA}(s) = \frac{27.8991(s+0.04443)}{s+0.02823}$$

Comparison of convergence of the fitness values for both approaches is shown in Figure 9. It is seen that the RSGA converges at the generation 4, whereas the SGA converges at the generation 14. Also, the fitness value attained by the RSGA is 3.75 which is comparatively higher than 3.7 of the SGA.

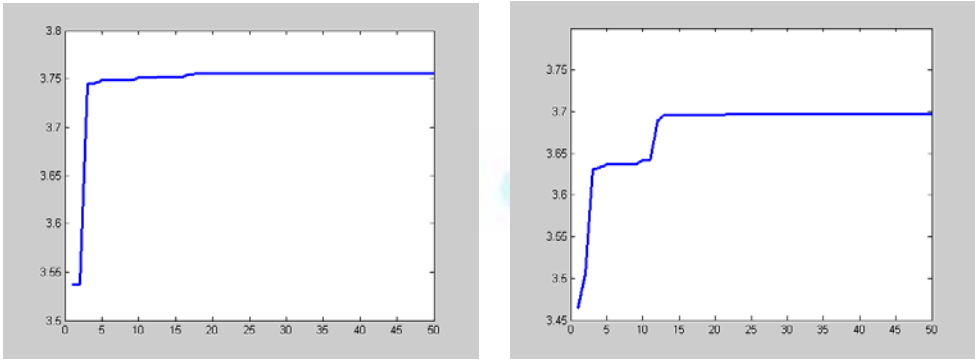
Table 6 summarizes performance of two designs. The RSGA controller yielded less overshoot and steady state error in transient responses than that of the SGA controller. Although the former system

consumed longer rise time (around 0.49%) than that obtained by the SGA, the RSGA controller significantly excels its opponent in performance.



(a) (b)

Figure 8. Comparison of structural variation for (a) RSGA, (b) SGA



(a) (b)

Figure 9. Convergence of the fitness value for (a) RSGA, (b) SGA

Table 6. Comparison of control system performance obtained by RSGA and SGA

Specificaton	Performance	
	SGA	RSGA
Max. Overshoot (%)	0.12635	0.04269
Rise Time (sec)	0.5058000	0.5083000
Steady State Error (%)	0.12635	0.03657

CONCLUSIONS

This chapter introduces the mechanism of a class of GAs, giving special emphasis on the RSGA, a variant of GAs and SGAs. The RSGA is capable of simultaneously dealing with the structure and parameter optimization problems based on the real coded chromosome scheme. As it adopts the same crossover and mutation for genetic operations, it performs better in computational efficiency and results in even evolution of offsprings. Novelty of this genetic model also lies in redundant genetic material and an adaptive activation mechanism. For demonstration, the genetic search approach in the designs of digital filters and robust controllers are introduced. The results show that this GA model exhibits better performance in pursing solutions while compared with the existing GAs.

References

Ashlock, D. 2006. Evolutionary Computation for Modeling and Optimization. Springer.

Balas, G. J., R. Chiang, A. Packard, and M. Safonov. 1995. Robust Control Toolbox. MathWorks, Natick, MA.

Bellas, F., J. A. Becerra, and R. J. Duro. 2009. Using promoters and functional introns in genetic algorithms for neuroevolutionary learning in non-stationary problems. Neurocomputing Journal 72:2134-2145.

Chen, B. S., and Y. M. Cheng. 1998. A structure-specified H_∞ optimal control design for practical applications: a genetic approach. IEEE Trans. Control System Technology 6:707-718.

Chen, J., E. Antipov, B. Lemieux, W. Cedeno, and D. H. Wood. 1999. DNA computing implementing genetic algorithms, in Proc. DIMACS Workshop on Evolution as Computation, 39-51.

Dasgupta, D., and D. R. Mcgregor. 1991. A structured genetic algorithm: the model and the first results. Res. Rep. IKBS-2-91, Strathclyde University,U.K.

Dasgupta, D., and D. R McGregor. 1992. Designing application-specific neural networks using the structured genetic algorithm, in Proc. Int. Conf. Combinations of Genetic Algorithms and Neural Networks, 87-96.

Dasgupta, D., and D. R. McGregor. 1992. Nonstationary function optimization using the structured genetic algorithm, in Proc. Int. Conf. Parallel Problem Solving from Nature, 145-154.

Etter, D. M., M. J. Hicks, and K. H. Cho. 1982. Recursive adaptive filter design using an adaptive genetic algorithm, in Proc. IEEE Int. Conf. ICASSP, 635-638.

Gen, M., and R. Cheng, 2000. Genetic Algorithms and Engineering Optimization. New York: Wiley.

Goldberg, D. E. 1991. Real-coded genetic algorithms, virtual alphabets and blocking. Complex Systems 5:139-167.

Green, M., and D. J. N. Limebeer. 1995. Linear Robust Control. NJ: Prentice-Hall, Englewood Cliffs.

Hlavacek, M., R. Kalous, and F. Hakl. 2009. Neural network with cooperative switching units with application to time series forecasting, in Proc. WRI World Congress on Computer Science and

Information Engineering, 676-682,

Holland, J. H. 1975. Adaptation in Natural and Artificial Systems. Cambridge: MIT Press, MA.

Jamshidi, M., R. A. Krohling, L. S. Coelho, and P. J. Fleming. 2002. Robust Control Systems with Genetic Algorithms. CRC Press.

Kaji, B., G. Chen, and H. Shibata. 2003. Design of reduced-order controller by using real-coded genetic algorithm, in Proc. Southeastern Symposium System Theory, 48-52.

Lai, C. C., and C. Y. Chang. 2004. A hierarchical genetic algorithm based approach for image segmentation, in Proc. IEEE Int. Conf. on Networking, Sensing and Control, 1284-1288.

Lin, C. L., and H. Y. Jan. 2005. Mixed/multiobjective PID control for a linear brushless DC motor: an evolutionary approach. Control and Intelligent Systems 33:75-86.

Liu, Y. W., C. W. Tsai, J. F. Lin, and C. L. Lin. 2007. Optimal design of digital IIR filters using real structured genetic algorithm, in Proc. Int. Conf. Computational Intelligence and Multimedia Applications, 493-497.

Tang, K. S., K. F. Man, S. Kwong, and Z. F. Liu. 1982. Design and optimization of IIR filter structure using hierarchical genetic algorithms. IEEE Trans. Ind. Electronics 45:481-487.

Tang, K. S., K. F. Man, and D. W. Gu. 1996. Structured genetic algorithm for robust H_∞ control system design. IEEE Trans. Ind. Electronics 43:575-582.

Tang, K. S., K. F. Man, S. Kwong, and Z. F. Liu. 1998. Minimal fuzzy memberships and rules using hierarchical genetic algorithms. IEEE Trans. Ind. Electronics 45:162-169.

Tsai, C. W., C. H. Huang, and C. L. Lin. 2009. Structure-specified IIR filter and control design using real structured genetic algorithm. Applied Soft Computing 9:1285-1295.

Shynk, J. J. 1989. Adaptive IIR filtering. IEEE ASSP Mag. 6:4-21.