THE SYNOPSIS OF SOFT COMPUTING

Duke Stephen Orok[1], Okwong Atte Enyenihi[2], and Emmanuel U. Oyo-Ita[3]

[1]Department of Computer Science, University of Cross River State, Calabar, Nigeria.
Email: orokduke2003@yahoo.com and orokduke2003@unicross.edu.ng

[2]Department of Computer Science, University of Cross River State, Calabar, Nigeria.
Email: atteokwong@unicross.edu.ng

[3]Department of Computer Science, University of Cross River State, Calabar, Nigeria.
Email: emmanueloyoita@unicross.edu.ng

Cite this article:

Duke, S. O., Okwong, A. E., Emmanuel, U. O. (2024), The Synopsis of Soft Computing.. British Journal of Computer, Networking and Information Technology 7(4), 47-57. DOI: 10.52589/BJCNIT-EIDOBTGJ

Manuscript History

Received: 12 Jul 2024

Accepted: 16 Sep 2024

Published: 9 Oct 2024

ABSTRACT: *The traditional method of problem-solving, known as hard computing, is limited in its ability to handle modern digital technology and real-world problems accurately. Soft computing, a newer paradigm, offers a more versatile approach by utilizing multi-valued logic and human knowledge to solve complex, nonlinear problems efficiently. Unlike hard computing, soft computing can handle imprecise data and uncertainty effectively. This methodology has been successfully applied across various sectors, including scientific, industrial, and medical fields, providing more accurate results. Soft computing is good for its contributions to revolutionizing problem-solving techniques, being tolerant of imprecision, uncertainty, and linguistic variables, and offering approximate solutions to intricate problems.*

KEYWORDS: Soft computing, Hard computing, Fuzzy logic, Artificial neural networks, Genetic algorithms, Expert systems.

INTRODUCTION

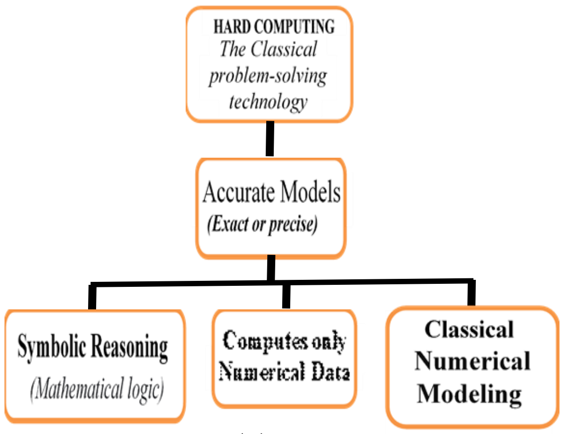
Hard computing and soft computing are two techniques in problem-solving. Modern control systems deal with complex issues that classical hard computing cannot precisely describe. Lotfi Zadeh coined the term "hard computing," which is the standard computing approach that requires an accurate analytical model. Hard computing is based on binary logic and predefined instructions, such as numerical analysis and brisk or swift systems, and employs both two-valued and crisp systems. Complex plants are difficult to control using the hard-computing paradigms because they are associated with imprecision, vagueness, ambiguity, and uncertainty. Soft computing methodologies can provide the most precise solutions to these complex problems. When solving complex problems, soft computing addresses imprecision, vagueness, partial truth, approximation, ambiguity, and uncertainty. Soft computing focuses on tolerance for partial truth, uncertainty, and imprecision to achieve low costs, robustness, and better rapport, with the human mind as the role model. The role model for soft computing is the human mind.

In reality, the human mind serves as the soft computing role model. Soft computing can accurately handle imprecision, resilience, uncertainty, partial truth, and approximation, making it the ideal candidate. Soft computing has several aspects that distinguish it as an intelligent control technique, including optimization, nonlinear programming, and decision-making assistance techniques. These characteristics have made soft computing very popular and sparked a lot of research interest from people from various backgrounds (Jang et al., 1997). Soft computing is the new problem-solving horizon, becoming the hottest problem-solving topic that is changing the computing landscape. Soft computing is not a single comprehensive computing technique. It consists of various correlated strategies that are complementary instead of conflicting with each other. Buckley and Hayashi (1994) state that the various strategies or components of soft computing can be integrated to create a hybrid technique to provide solutions to given complex problems. The most important components of soft computing are Neural Artificial Networks (ANNs), Probabilistic Reasoning (PR), Fuzzy Logic (FL), Neurocomputing and Genetic Algorithms (GAs).

Hard computing methodology struggles to manage modern machinery complexity due to challenges with non-linear and time-variant plants. Soft computing methodology is preferred for controlling complex systems by leveraging uncertainty and imprecision in decision-making.

Hard Computing versus Soft Computing Paradigms

Hard computing and soft computing are dominant problem-solving techniques. Hard computing relies on traditional math, using numbers for precise models and accurate solutions. Soft computing, an evolution from hard computing, handles nonlinear problems with fuzzy elements using numbers and linguistics for approximations. Figure 1 shows the activities of these techniques.


HARD COMPUTING
The Classical problem-solving technology
Accurate Models
(Exact or precise)
Symbolic Reasoning
(Mathematical logic)
Computes only Numerical Data
Classical Numerical Modeling

(a) **(b)**

Figure 1: Hard computing versus Software computing (Duke, 2024)

Figure 1(a) presents the classical hard computing problem-solving method, utilizing accurate models for solving simple and complex problems through symbolic reasoning and numerical modeling, computing numerical data perfectly. In contrast, figure 1(b) illustrates the soft computing problem-solving approach using approximate models for complex problem solving with both numerical and linguistic data. Table 1 details distinctions between the two methodologies.

Table 1: Soft computing versus Hard computing

Soft computing	**Hard computing**
(i) Stochastically in nature, hence the best candidate for solving real-world problems).	Deterministic in nature cannot solve real-world problems with accuracy.
(ii) Employs fuzzy logic (multi-valued) to solve problems.	Employs two-valued logic (binary) to solve problems
(iii) Accepts uncertainty and approximation, using the human brain as a model for problem-solving. When problems are solved at different times, they may produce different answers or results.	Intolerance for imprecision, uncertainty, and approximations.

(v) Can evolve its own program (learn itself) to compute.	Cannot evolve its own program. Require programs to be written to compute a particular problem.
spends much less computation time	spends much more computation time
(vi) Programs are not linked to hardware such as database and spreadsheet directly	Programs are linked or connected to hardware.
(vii) Is dependent on fuzzy logic, neural networks, probabilistic reasoning.	Is dependent on binary logic, crisp inputs, crisp systems, numerical analysis.
(viii) Robustness (can cope with errors during the time of execution as well as with erroneous inputs).	Not robust (cannot cope with errors during execution time as well as erroneous inputs). Requires accuracy.
(ix) Gives approximate answers or solutions	Gives exact answers or solutions.
(x) Robustness (can cope with errors during the time of execution as well as with erroneous inputs).	Not robust (cannot cope with errors during execution time as well as erroneous inputs). Requires accuracy.
(xi) Deals with noisy data	Deals with noisy data, but exact data
(xii) Is viewed as computation intelligence	Is viewed as conventional intelligence

Constituents of Soft Computing

Driankov et al. (1993) explain that complex control problems have been solved by means of fuzzy control techniques over the past few decades. Furthermore, fuzzy logic paradigms or concepts are being used to solve a variety of instrumental-based problems (Russo, 1996). Many people today use the newer concept of soft computing, or neural networks, to solve complex automatic control and servo-based problems (Gao & Ovaska, 1999). Soft computing has a wide range of applications, including pattern recognition (Pedrycz, 1990), design and manufacturing (Roy et al., 1998), signal processing (Cichocki and Unbehauen, 1998), heavy current systems (Dote & Hoft, 1998), intelligent speech recognition (Komori, 1992), and communications (Yuhas, 1994), among others.

This paper provides a synopsis of the soft computing paradigms and explains some of their most commonly used components in solving complex problems, including GAs, ANNs, FL, and ESs.

ANNs

ANN is a complex information processing system composed of interconnected neurons that manipulates information through dynamic state responses to external inputs. This emerging model, attracting researchers from various disciplines, is a rapidly growing field in fields like artificial intelligence, statistics, and cognitive psychology.

ANNs are information processing systems inspired by the biological nervous system and brain. They consist of interconnected networks of artificial neurons, each performing a small operation. ANNs are designed for specific applications like stock market prediction, pattern recognition, data recognition, security, image compression, and weather prediction. ANNM aims to bring computers closer to human brains.

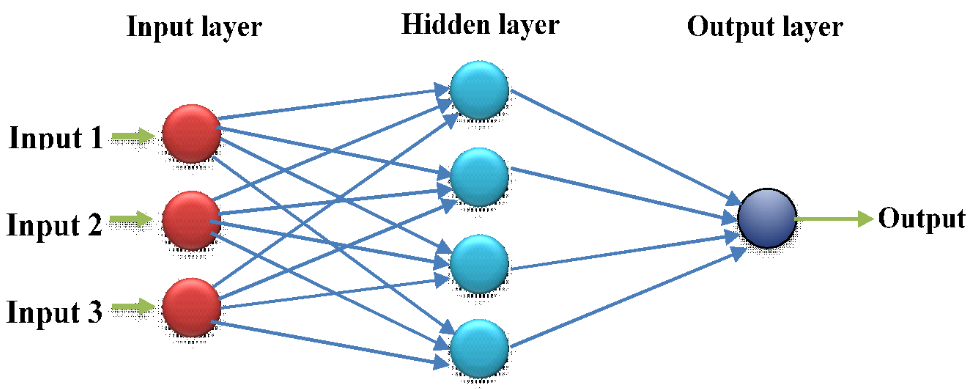


Figure 3: A three inputs single layer, one hidden layer, and one neuron output ANNs schematic

ANNs are modeling tools used to model non-linear statistical data and complex relationships between inputs and outputs. They are similar to connectionism in biological neural networks, but perform functions in units. Connectionism studies human cognition using ANNs to describe mental phenomena in neuron-like units.

Training the ANNs

ANNs are trained to produce desired outputs from known inputs by feeding teaching patterns and allowing the network to evolve its weighting function based on defined learning rules, which can be supervised or unsupervised. Most ANNs utilize the backpropagation algorithm for network training (Goh, 1995; Saerens, 2002).

Supervised learning involves an ANN where inputs are given and output patterns are matched, resulting in known outcomes for specific inputs. In contrast, unsupervised learning trains the network's output to respond to input patterns, resulting in unknown outcomes for specific inputs. Both approaches are crucial in understanding the performance of ANNs under investigation.

Areas of Application ANNs

ANNs have a broad range of applications. They are powerful models with diverse applications across various fields such as medicine, security, banking, agriculture, image processing, character recognition, forecasting, defense, government, stock exchange prediction, traveling salesman problem, and engineering.

Advantages of ANNs

NNs offer several advantages that make them highly suitable for solving specific problems and situations.

1. ANNs are crucial in real-life as they can learn and model complex, non-linear relationships between inputs and outputs.
2. ANNs can learn from initial inputs and relationships, deducing unseen relationships from unseen data, enabling the model to generalize and predict unseen data.
3. ANNs, unlike other prediction models, do not impose any restrictions on input variables, such as distribution. They can better model heteroskedasticity, high-volatility data with inconstant or changeable variance, as long as they can learn hidden relationships within the data without imposing fixed relationships, unlike other techniques that may impose restrictions on input variables.
4. Bone and Crucianu (1996) demonstrated that Artificial Neural Networks (ANNs) can successfully predict and forecast applications.
5. ANNs can handle incomplete data sets and knowledge, and after training, they can generate outputs even with incomplete information.
6. ANNs store information on the entire network, not just in a database, ensuring that a lack of information doesn't hinder network functionality.
7. ANNs have fault tolerance, meaning that corrupt cells in an ANN do not prevent it from generating output.
8. ANNs undergo gradual corruption over time, undergoing relative degradation, but their problems do not immediately corrode.
9. ANNs are capable of replicating machine learning by learning about events and making decisions based on similar events.
10. ANNs possess parallel processing capabilities, numerical strength, and can perform multi-task or multi-job simultaneously.

Disadvantages of ANNs

ANNs have certain disadvantages that make them unsuitable for solving specific problems and situations.

1. The lack of methodologies for training and verifying Artificial Neural Networks (ANNs) as they are not universally applicable problem-solving tools.
2. The preciseness of ANN results is dependent on the data used for the investigation.
3. There may be a need for extensive training in complex ANN systems.
4. ANNs are hardware-dependent, requiring processors with parallel processing powers, making the realization of the equipment dependent.
5. ANNs' unexplained behaviors, which lack explanation for why, how, or when they produce probing solutions, significantly reduce the credibility and sureness of the network.
6. ANNs' unexplained behaviors, which lack explanation for why, how, or when they produce probing solutions, significantly reduce the credibility and sureness of the network.
7. ANNs struggle to display problems to the network, requiring translation into numerical data before introduction. Network performance directly influences display mechanism performance, influenced by user ability.

GAs

GAs, a key component of AI and fuzzy computing, mimic natural selection to solve optimization problems in real-world scenarios. They are machine learning models that use evolution to find optimal solutions.

Areas of Application of GAs

Genetic algorithms are a form of artificial intelligence and fuzzy computing employed for solving real-life optimization problems by mimicking natural selection. Inspired by evolution, genetic algorithms are used in numerous fields like biomedical engineering, control engineering, design, and DNA analysis to find optimal solutions efficiently.

The Processes in GAs

Genetic algorithms have the following basic processes:

- Initialization Process: This is where we create an initial population randomly.
- Evaluation Process: This is where each member of the population is evaluated, and the fitness of the individuals is assessed based on how well they fit the desired requirements.
- Selection Process: This is where only the ones that fit the desired requirements are reselected.

- Crossover Process: This is where new individuals are created by combining the best aspects of the existing individuals.

ESs

An ES replicates human decision-making through Artificial Intelligence. ESs are designed by experts to solve problems by using IF-THEN rule statements instead of procedural codes. Expert systems are adaptable to external factors, capable of decision-making. They can replace or support humans in various fields like engineering, law, medical diagnosis, robotics, and financial decisions. They offer valuable applications in different areas.

Components of ESs

Components of ESs: Knowledge Base (KB), Working Memory (WM), Inference Engine (IE), and User Interface (UI) for functionality.

i. **KB:** KB is crucial in ESs, storing intelligence through IF-THEN-ELSE constructs. It houses rules guiding knowledge use and problem-solving. ESs gather knowledge from sensors, training, and adapt to efficiently handle new problems.

ii. **WM:** WM is an ES system's Black Box for tracking problems/results. It describes the current problems and records the intermediate results.

iii. **IE:** IE is the brain of ESs, inferring results from user input and KB, providing reasoning methodology.

iv. **UI:** The User Interface serves as the bridge between users and systems, employing Natural Language Processing, menus, and graphics for communication. Positioned between the KB and users, the UI facilitates problem-oriented interactions by processing language. It plays a crucial role in decision-making, complying with conditions and requirements to provide solutions. ESs are commonly created with symbolic and procedural programming languages like Clips, Prolog, OPS-5, Python, and Lisp.

Types of Expert System

Expert systems comprise five primary types:

i. Frame-based expert systems

ii. Neural expert systems

iii. Rule-based expert systems

iv. Fuzzy expert systems

v. Neuro-fuzzy expert systems

Expert Systems' Problems and Failures

The significant problems that ESs run into are:

1. **ESs are brittle:**

i. ESs struggle with rules-bending problems

ii. Weakness or groundlessness in certain cases hinders their effectiveness.

iii. Inability to apply experience details and locate related cases hinders training

iv. Inability to retain cases cripples their learning capacity.

2. **Problem of database size:** Technically, the size of the database as well as using it efficiently are problems, thus:

i. An expert system containing several thousand rules requires a very powerful control program to produce any conclusions within a reasonable time.

ii. An ES containing enormous amounts of information in the working memory slows down things, except we have very good indexing and search systems to deal with the situation.

iii. As the number of rules in the ES increases, the conflicts also become larger, requiring good conflict resolution algorithms if the expert systems are to be usable.

3. **Problem of responsibility:** A doctor-designed system administering drugs based on patient needs, requiring diagnosis, could lead to harm if the wrong medicine is taken. The responsibility for the mishap lies with the health authority, the doctor, or the supplier of the ES.

4. **Problem of gathering the rules in the system:** Obtaining rules for ESs is challenging due to high costs and reluctance of human experts to document their decision-making process extensively. Despite following logic, translating their paths into IF...THEN rules can be exceedingly hard or even unachievable.

5. **Problem of Knowledge Acquisition:** There exists a serious shortage of human power to provide the required knowledge in ES development.

FL

Figure 2 depicts a typical FL system's architecture, which includes four main interfaces: fuzzification, inference engine, fuzzy rule base, and defuzzification. FL systems process both numerical and linguistic input data by transforming crisp inputs into fuzzy linguistic values. The inference engine processes fuzzy inputs and rules to generate fuzzy outputs. Defuzzification converts fuzzy outputs back to crisp values. Ultimately, FL systems effectively capture the imprecise nature of human knowledge.

One of the most significant advantages of FL is that it gives practical methods for creating nonlinear control systems, which are difficult to design and stable using traditional methods.

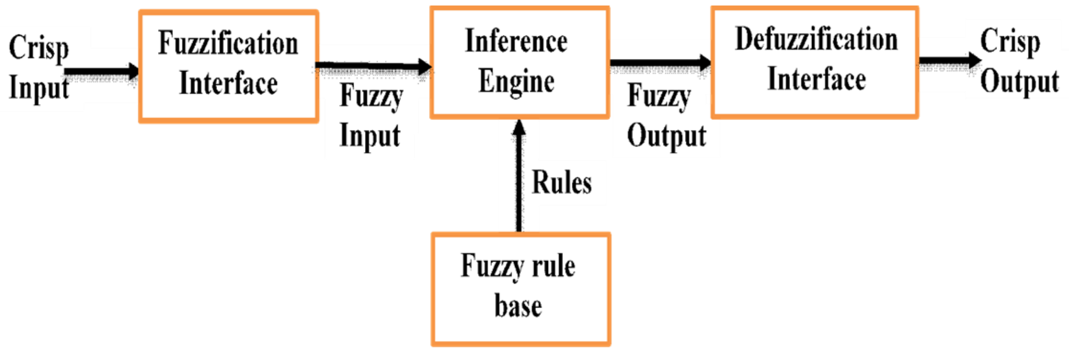


Figure 2: A Typical Architecture of Fuzzy Logic

CONCLUSIONS AND THE FUTURE OF SOFT COMPUTING

As computer technologies advance, intelligent systems and soft computing paradigms become increasingly important for making complex decisions in real life. With high-speed processing power and large storage available in research centers and universities, these systems are used in various fields such as learning, scheduling, vehicle routing, image processing, parallelization, neural networks, and economics.

The importance of using the soft computing method in building intelligent systems, particularly in the realm of the Internet of Things (IoT), has significantly increased in recent times.

FL, ANN, GA, and ES are widely used in everyday domestic appliances like washing machines, cookers, fridges, sewing machines, car washing, cleaning machines, and kitchen blenders. They also have applications in sciences, industries, governments, military, aviation, and commerce, with growth expected.

REFERENCE

Bone, R., & Crucianu, M. (1996.) *Multi-step-ahead Prediction with Neural Networks.* A review. Publication de l'équipe RFAI. 9 emes Rencontre 689; April 1996.

Buckley, J. J. & Hayashi, Y. (1994.) Neural Networks. A survey. Fuzzy Sets and Systems 1994;66: 1-13.

Cichocki, A., & Unbehauen R. (1998.) *Neural Networks for Optimization and Signal Processing*. West Sussex: UK. John Wiley & Sons.

Dote, Y., & Hoft, R. G. (1998) *Intelligent Control*: *Power Electronic Systems*. Oxford: UK. Oxford University Press.

Duke, S. O. (2024) Praise for Soft Computing. Journal of Science, Engineering and Technology, Vol. 10 (1), March 2023: pages 98-105

Driankov, D., Reinfrank, M., & Hellendoorn, H. (1993). *An Introduction to Fuzzy Control.* Berlin: Germany.Springer; 1993.

Gao, X. Z., & Ovaska, S. J. (1999.) Friction compensation in servo motor systems using neural networks. in *Proc. IEEE Midnight-Sun Workshop on Soft Computing Methods in industrial applications.* Kuusamo: Finland; June 1999.

Goh, A.T.C. (1995.) Back-propagation neural networks for modeling complex systems, *Artificial Intelligence in Engineering 1995;*9(3):143-151.

Holland, J. H. (1992.) Adaptation in Natural and Artificial Systems. *An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. 2nd ed*. Cambridge: MA. MIT Press;1992.

Jang, J. S. R., Sun C.T., & Mizutani, E. (1997.) Neuro-Fuzzy and Soft Computing, A Computational Approach to Learning and Machine Intelligence. Upper Saddle River, NJ: Prentice-Hall; 1997.

Komori, Y. (1992.) A neural fuzzy training approach for continued speech recognition improvement. in *Proc. International Conference on Acoustics, Speech, and Signal Processing.* San Francisco: CA; 1992: 405-408.

Pedrycz, W. (1990.) Fuzzy sets in pattern recognition: Methodology and methods, *Pattern Recognition 1990;* vol. 23. no. 1: 121-146.

Roy, R. Furuhashi, T, & Chawdhry P. K. (1998.) *Advances in Soft Computing: Engineering Design and Manufacture*. London.

Russo, F. (1996.) Fuzzy systems in instrumentation: Fuzzy signal processing. *IEEE Trans Instrumentation and Measurement.* vol.45. no.2.

Saerens, M. (2002.) Neural controller based on back-propagation algorithm. *IE Proc. On Radar and Signal Processing 2002;* 138(1): 55-62.

Whitley, D. A. (1994.) *Genetic algorithm tutorial. Statistics and Computing* 1994; 4(2): 65-85.

Yuhas, B., & Ansari, N. (1994.) *Neural Networks in Telecommunications.* Boston: MA. Kluwer Academic; 1994.