Scientific Journal Impact Factor (SJIF): 1.711

e-ISSN: 2349-9745
p-ISSN: 2393-8161

International Journal of Modern Trends in Engineering and Research
www.ijmter.com

Soft Computing: A survey

Lukesh M. Barapatre[1], Anand Sharma[2]
[1]Department of Information Technology, DMIETR, Wardha
[2]Department of Information Technology, SGMCE, Shegaon

Abstract – Soft Computing is the fusion of methodologies that were designed to model and enable solutions to real world problems, which are not modeled or too difficult to model, mathematically. Soft computing is a consortium of methodologies that works synergistically and provides, in one form or another, flexible information processing capability for handling real-life ambiguous situations. Its aim is to exploit the tolerance for imprecision, uncertainty, approximate reasoning and partial truth in order to achieve tractability, robustness and low-cost solutions. The guiding principle is to devise methods of computation that lead to an acceptable solution at low cost, by seeking for an approximate solution to an imprecisely or precisely formulated problem.Soft Computing (SC) represents a significant paradigm shift in the aims of computing, which reflects the fact that the human mind, unlike present day computers, possesses a remarkable ability to store and process information which is pervasively imprecise, uncertain and lacking in categoricity. At this juncture, the principal constituents of Soft Computing (SC) are: Fuzzy Systems (FS), including Fuzzy Logic (FL); Evolutionary Computation (EC), including Genetic Algorithms (GA); Neural Networks (NN), including Neural Computing (NC); Machine Learning (ML); and Probabilistic Reasoning (PR). In this paper we focus on fuzzy methodologies and fuzzy systems, as they bring basic ideas to other SC methodologies

Index Terms – Soft Computing, Machine Learning, Fuzzy logic

I. INTRODUCTION

Soft Computing is a term which to refer to problems in computer science whose solutions are unpredictable, uncertain and between 0 and 1. Soft Computing became a formal area of study in Computer Science in the early 1990s. Earlier computational approaches could model and precisely analyze only relatively simple systems. More complex systems arising in biology, medicine, the humanities, management sciences, and similar fields often remained intractable to conventional mathematical and analytical methods. That said, it should be pointed out that simplicity and complexity of systems are relative, and many conventional mathematical models have been both challenging and very productive. Soft computing deals with imprecision, uncertainty, partial truth, and approximation to achieve practicability, robustness and low solution cost. As such it forms the basis of a considerable amount of machine learning techniques. Recent trends tend to involve evolutionary and swarm intelligence based algorithms and bio-inspired computation.Soft computing is a consortium of methodologies that works synergistically and provides, in one form or another, flexible information processing capability for handling real-life ambiguous situations. Its aim is to exploit the tolerance for imprecision, uncertainty, approximate reasoning and partial truth in order to achieve tractability, robustness and low-cost solutions**.** Soft computing differs from conventional (hard) computing. Unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth and approximation. In effect, the role model for soft computing is the human mind. Soft Computing is basically optimization technique to find solution of problems which are very hard to answer.

In real world, we have many problems which we have had no way to solve analytically, or problems which could be solved theoretically but actually impossible due to its necessity of huge resources and/or enormous time required for computation. For these problems, methods inspired by nature sometimes work very efficiently and effectively. Although the solutions obtained by these methods do not always equal to the mathematically strict solutions, a near optimal solution is sometimes enough in most practical purposes. These biologically inspired methods are called Soft Computing,

II. FUZZY SYSYEM

The concept of Fuzzy Logic (FL) was conceived by Lotfi Zadeh, a professor at the University of California at Berkley, and presented not as a control methodology, but as a way of processing data by allowing partial set membership rather than crisp set membership or non-membership. This approach to set theory was not applied to control systems until the 70's due to insufficient small-computer capability prior to that time. Professor Zadeh reasoned that people do not require precise, numerical information input, and yet they are capable of highly adaptive control. If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement. Unfortunately, U.S. manufacturers have not been so quick to embrace this technology while the Europeans and Japanese have been aggressively building real products around it.

A. WHAT IS FUZZY LOGIC?

In this context, FL is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software, or a combination of both. FL provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. FL's approach to control problems mimics how a person would make decisions, only much faster.

B. HOW DOES FL WORK?

FL requires some numerical parameters in order to operate such as what is considered significant error and significant rate-of-change-of-error, but exact values of these numbers are usually not critical unless very responsive performance is required in which case empirical tuning would determine them. For example, a simple temperature control system could use a single temperature feedback sensor whose data is subtracted from the command signal to compute "error" and then time-differentiated to yield the error slope or rate-of-change-of-error, hereafter called "error-dot". Error might have units of degs F and a small error considered to be 2F while a large error is 5F. The "error-dot" might then have units of degs/min with a small error-dot being 5F/min and a large one being 15F/min. These values don't have to be symmetrical and can be "tweaked" once the system is operating in order to optimize performance. Generally, FL is so forgiving that the system will probably work the first time without any tweaking.

III. EVOLUTIONARY COMPUTATION

Evolutionary computation, offers practical advantages to the researcher facing difficult optimization problems. These advantages are multi-fold, including the simplicity of the approach, its robust response to changing circumstance, its flexibility, and many other facets. The evolutionary approach can be applied to problems where heuristic solutions are not available or generally lead to unsatisfactory results. As a result, evolutionary computations have received increased interest, particularly with regards to the manner in which they may be applied for practical problem solving.

In nature, evolution is mostly determined by natural selection or different individuals competing for resources in the environment. Those individuals that are better are more likely to survive and propagate their genetic material.

The encoding for genetic information (genome) is done in a way that admits asexual reproduction which results in offspring that are genetically identical to the parent. Sexual reproduction allows some exchange and re-ordering of chromosomes, producing offspring that contain a combination of information from each parent. This is the recombination operation, which is often referred to as crossover because of the way strands of chromosomes cross over during the exchange. The diversity in the population is achieved by mutation. Evolutionary algorithms are ubiquitous nowadays, having been successfully applied to numerous problems from different domains, including optimization, automatic programming, machine learning, operations research, bioinformatics, and social systems. In many cases the mathematical function, which describes the problem is not known and the values at certain parameters are obtained from simulations. In contrast to many other optimization techniques an important advantage of evolutionary algorithms is they can cope with multi-modal functions. Usually grouped under the term evolutionary computation or evolutionary algorithms, we find the domains of genetic algorithms, evolution strategies evolutionary programming and genetic programming. They all share a common conceptual base of simulating the evolution of individual structures via processes of selection, mutation, and reproduction. The processes depend on the perceived performance of the individual structures as defined by the problem. A population of candidate solutions (for the optimization task to be solved) is initialized. New solutions are created by applying reproduction operators (mutation and/or crossover). The fitness (how good the solutions are) of the resulting solutions are evaluated and suitable selection strategy is then applied to determine which solutions will be maintained into the next generation. The procedure is then iterated and is illustrated in

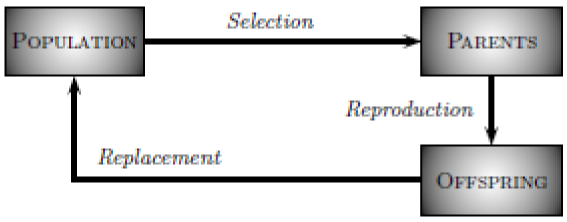

Fig 1:Flow chart of an evolutionary algorithm

A. Advantages of evolutionary algorithms

Following are some of the advantages of using evolutionary algorithms while compared to other global optimization techniques

1. Evolutionary algorithm performance is representation independent in contrast to other numerical techniques, which might be applicable for only continuous values or other constrained sets.

2. Evolutionary algorithms offer a framework such that it is comparably easy to incorporate prior knowledge about the problem. Incorporating such information focuses the evolutionary search, yielding a more efficient exploration of the state space of possible solutions..
3. Evolutionary algorithms can also be combined with more traditional optimization techniques. This may be as simple as the use of a gradient minimization after primary search with an evolutionary algorithm (e.g. fine tuning of weights of an evolutionary neural network) or it may involve simultaneous application of other algorithms (e.g. hybridizing with simulated annealing or Tabu search to improve the efficiency of basic evolutionary search).
4. The evaluation of each solution can be handled in parallel and only selection (which requires at least pair-wise competition) requires some serial processing. Implicit parallelism is not possible in many global optimization algorithms like simulated annealing and Tabu search.
5. Traditional methods of optimization are not robust to the dynamic changes in the problem of the environment and often require a complete restart in order to provide a solution (e.g. dynamic programming). In contrast, evolutionary algorithms can be used to adapt solutions to changing circumstance.
6. Perhaps, the greatest advantage of evolutionary algorithms comes from the ability to address problems for which there are no human experts. Although human expertise should be used when it is available, it often proves less than adequate for automating problem solving routines.

IV. GENETIC ALGORITHMS (GA)

A typical flowchart of a genetic algorithm is depicted in Figure 2. One iteration of the algorithm is referred to as a *generation*. The basic GA is very generic, and there are many aspects that can be implemented differently according to the problem (e.g. representation of solution (chromosomes), type of encoding, selection strategy, type of crossover and mutation operators, etc.).

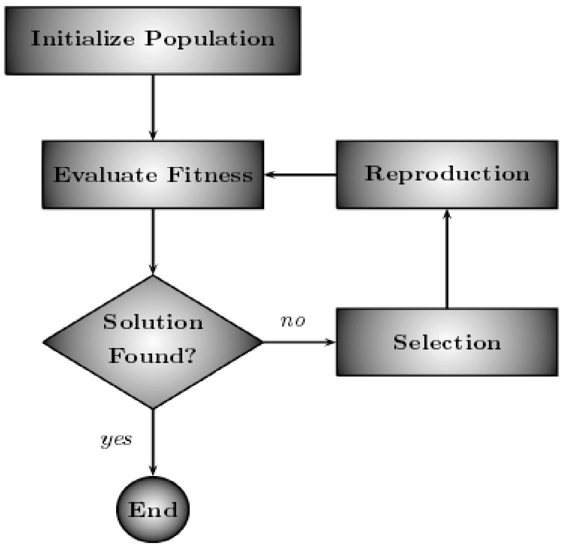

Fig 2:Flow chart of basic genetic algorithm iteration

In practice, GAs are implemented by having arrays of bits or characters to represent the chromosomes. The individuals in the population then go through a process of simulated evolution. Simple bit manipulation operations allow the implementation of crossover, mutation and other operations. The

number of bits for every gene (parameter) and the decimal range in which they decode are usually the same but nothing precludes the utilization of a different number of bits or range for every gene.
GA starts off with population of randomly generated chromosomes, each representing a candidate solution to the concrete problem being solved and advances towards better chromosomes by applying genetic operators based on the genetic processes occurring in nature. So far, GAs had a great measure of success in search and optimization problems due to their robust ability to exploit the information accumulated about an initially unknown search space. Particularly GAs specialize in large, complex and poorly understood search spaces where classic tools are inappropriate, inefficient or time consuming. As mentioned, the GA's basic idea is to maintain a population of chromosomes. This population evolves over time through a successive iteration process of competition and controlled variation. Each state of population is called generation. Associated with each chromosome at every generation is a fitness value, which indicates the quality of the solution, represented by the chromosome values.
Based upon these fitness values, the selection of the chromosomes, which form the new generation, takes place. Like in nature, the new chromosomes are created using genetic operators such as crossover and mutation.

A. Mechanism of GA

The fundamental mechanism consists of the following stages which are also described in figure 3.
1. Generate randomly the initial population.
2. Select the chromosomes with the best fitness values.
3. Recombine selected chromosomes using crossover and mutation operators.
4. Insert offspring into the population.
5. If a stop criterion is satisfied, return the chromosome(s) with the best fitness. Otherwise, go to Step 2.
In GA, the population is defined to be the collection of individuals. A population is a generation that undergoes under changes to produce new generation.
Like nature, GAs has also collection of several members to make population healthy. A chromosome that is a collection of genes is correspondence to individual of population. Each individual chromosome represents a possible solution to the optimization problem. The dimension of the GA refers to the dimension of the search space which equals the number of genes in each chromosome.

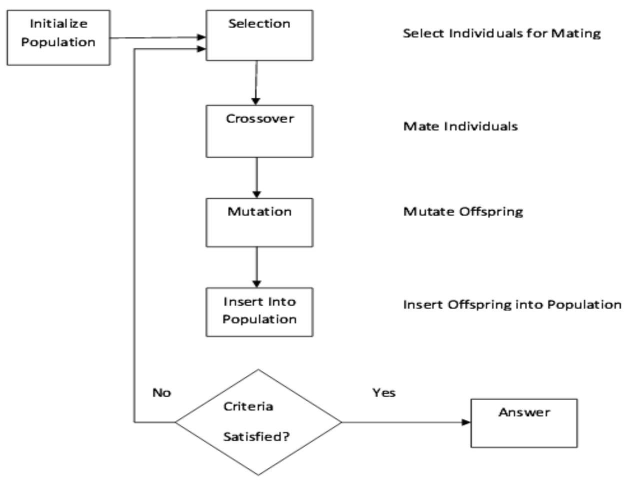

Figure 3: Fundamental Mechanism of Simple Genetic Algorithm

V. NEURAL NETWORKS (NNS)

There are millions of very simple processing elements or neurons in the brain, linked together in a massively parallel manner. This is believed to be responsible for
the human intelligence and discriminating power. Neural Networks are developed to try to achieve biological system type performance using a dense interconnection of simple processing elements analogous to biological neurons.
Neural Networks are information driven rather than data driven. Typically, there are at least two layers, an input layer and an output layer. One of the most common networks is the Back Propagation Network (BPN) which consists of an input layer, and an output layer with one or more intermediate hidden layer. Neural Networks are trained to perform a particular function by adjusting the values of the connections (weights) between elements using a set of examples before they can be employed to the actual problem. Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output.The method used to generate the examples to train the network and the training algorithm employed has a significant impact on the performance of the neural network-based model. One of the training algorithms used is the Back- Propagation (BP) algorithm. This algorithm aims to reduce the deviation between the desired objective function value and the actual objective function value.

VI. PARTICLE SWARM OPTIMIZATION (PSO)

Although GAs provides good solution but they not keep information about the best solution in the whole community. This strategy extends search by the introduction of memory. In this optimization, along with the local best solution, a global best solution is also stored somewhere in the memory, so that all particles not trapped into local optima but moves to global optima. PSO is an algorithm developed by Kennedy and Eberhart that simulates the social behaviors of bird flocking or fish schooling and the methods by which they find roosting places, foods sources or other suitable habitat. The algorithm maintains a population potential where each particle represents a potential solution to an optimization problem. The PSO algorithm works by simultaneously maintaining several candidate solutions in the search space. During each iteration of the algorithm, each candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution. Each candidate solution can be thought of as a particle “flying” through the fitness landscape finding the maximum or minimum of the objective function. Initially, the PSO algorithm chooses candidate solutions randomly within the search space. The initial state of a four-particle PSO algorithm seeking the global maximum in a one-dimensional search space. The search space is composed of all the possible solutions along the x-axis. The curve denotes the objective function. PSO algorithm has no knowledge of the underlying objective function. hus has no way of knowing if any of the candidate solutions are near to or far away from a local or global maximum. PSO algorithm uses the objective function to evaluate its candidate solutions and operates upon the resultant fitness values.

Each particle maintains its position, composed of the candidate solution and its evaluated fitness and its velocity. Additionally, it remembers the best fitness value it has achieved thus far during the operation of the algorithm, referred to as the individual best fitness and the candidate solution that achieved this fitness, referred to as the individual best position or individual best candidate solution. Finally, PSO algorithm maintains the best fitness value achieved among all particles in the swarm. It is called the global best fitness. The candidate solution that achieved this fitness is called the global best position or global best candidate solution.

VII. CONCLUSION

In this paper we studied the various soft computing techniques such as Fuzzy logic, Evolutionary algorithm, Genetic algorithm, Neural Networks and PSO.FL was conceived as a better method for sorting and handling data but has proven to be a excellent choice for many control system applications since it mimics human control logic. It can be built into anything from small, hand-held products to large computerized process control systems. It uses an imprecise but very descriptive language to deal with input data more like a human operator.

REFERNCES

[1] "Europe Gets into Fuzzy Logic" (Electronics Engineering Times, Nov. 11, 1991).
[2] "Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh", ed. R.R. Yager et al. (John Wiley, New York, 1987).
[3] "U.S. Loses Focus on Fuzzy Logic" (Machine Design, June 21, 1990).
[4] Koza, J.R., ”Genetic Programming II”, A Bradford Book, MIT Press, Cambridge, 1994.
[5]Shanahan, J.G., Soft Computing for Knowledge Discovery, Kluwer Acad.Publ., Boston /Dordrecht /London, 2000
[6] Barber, J. C. 1995. "Genetic Algorithms as Tools for Optimization," *Risks and Rewards*, December.
[7] Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*. Clarendon Press.
[8] Neural Network Learning and Expert Systems,Cambridge, MA:MIT Press, 1994.
[9] I. A. Taha and J. Ghosh, “Symbolic interpretation of artificial neural networks,”IEEE Trans. Knowl. Data Eng., vol. 11, pp. 448–463, 1999.
[10] Abraham A (2004) Meta-Learning Evolutionary Artificial Neural Networks, Neurocomputing Journal, Elsevier Science, Netherlands, 56c, pp. 1–38
[11] Ahuja RK, Orlin JB, and Tiwari A (2000) A greedy genetic algorithm for the quadratic assignment problem, Computers and Operations Research, 27, 917–934
[12] Aruldoss AVT and Ebenezer JA (2004) Hybrid PSO-SQP for economic dispatch with valve-point effect, Electric Power Systems Research, 71(1), pp. 51–59

.