Chapter 2
An Overview of Computational Intelligence Algorithms

This chapter provides an overview of selected computational intelligence algorithms, which will be required to understand the rest of the book. It begins with a review of fuzzy sets and logic, and would gradually explore swarm and evolutionary algorithms, and neural nets. The coverage on swarm and evolutionary algorithms include Genetic Algorithm, Particle Swarm Optimization Bio-geography Based Optimization and Differential Evolution algorithm. Supervised, unsupervised and reinforcement learning algorithms will be outlined under neural nets. The chapter ends with scope of applications of computational intelligence algorithms in call admission.

2.1 Introduction

The chapter introduces the foundations of computational intelligence techniques in a nutshell. Computational intelligence is a vast family of knowledge comprising of several models and techniques, including the logic of fuzzy sets, neurocomputing, and swarm and evolutionary computation. The logic of fuzzy sets provides a frame work for uncertainty management in complex reasoning systems. Neural nets offer automatic techniques of machine learning and pattern recognition. Evolutionary algorithms are widely being used for intelligent search, optimization and machine learning. Swarm algorithms are nature-inspired techniques capable of imitating nature by judiciously selecting their power of natural optimization in identifying food sources. Researchers are taking keen interest to develop newer bio-inspired optimization algorithms by imitating the behavior of lower level creatures such as ants, bees and swaps.

The chapter begins with logic of fuzzy sets. It introduces membership functions and fuzzy relations with special reference to implication relations. It also reviews fuzzy reasoning. Principles of machine learning are introduced through artificial neural networks. The chapter provides the basis of learning in biological nervous system, and its electrical equivalent model, that too stems from its biological counterpart. It also briefly overviews supervised and unsupervised learning, the basic two learning policies that humans normally employ in their natural learning process. The later part of the chapter overviews a few swarm and evolutionary algorithms, covering Particle Swarm Optimization (PSO), Biogeography Based Optimization (BBO) and Genetic Algorithm and Differential Evolution algorithm (DE).

S. Ghosh and A. Konar: CAC in a Mobile Cellular Network, SCI 437, pp. 63–94.
springerlink.com

2.2 A Review of Fuzzy Sets and Logic

In a conventional set, the condition defining the set boundaries is very rigid. For example, consider a universal set AGE, OLD, VERY OLD, YOUNG, CHILD and BABY are subsets of the universal set AGE. The conventional approach to define these sets is illustrated below:

$$BABY = \{age \in AGE: 0\ year \leq age < 1year\},$$

$$CHILD = \{age \in AGE: 1\ year \leq age \leq 10years\},$$

$$YOUNG = \{age \in AGE: 19\ years \leq age \leq 40years\},$$

$$OLD = \{age \in AGE: 60\ years \leq age < 80years\},$$

$$and\quad VERY\ OLD = \{age \in AGE: 80\ years \leq age < 120\ years\}.$$

In the above definitions age is a variable that may presume any value in the range [0, 120] years. It is clear from the definition that the boundary of each set is distinct. Thus an age=11 months 29 days is a member of the set BABY, but once it is 1 year it falls in the set CHILD. Thus there is a sharp demarcation in the boundary definition of the sets BABY and CHILD at age=1 year. Measurements in a real world system being highly imprecise, such a sharp demarcation of 2 set boundaries may cause a wrong allocation of the members to a given set.

Another characteristic of a conventional set includes assignment of a grade of membership 1 to all its members and 0 to all its non-members. The following connotation is used to describe that the membership of an element x in a set A is 1, and the membership of a non-element y in the set A is 0.

$$\mu_A(x) = 1 \tag{2.1}$$

$$\mu_A(y) = 1 \tag{2.2}$$

A fuzzy set extends the binary membership: {0,1} of a conventional set to a spectrum in the interval of [0, 1]. Further, unlike a conventional set, all elements of the universal set U are members of a given set A. Thus for each element $x \in U$,

$$0 \leq \mu_A(x) \leq 1\ . \tag{2.3}$$

It needs mention here that as all elements of a universal set U are members of a given fuzzy set A, therefore, 2 fuzzy sets A and B may have an overlap in the boundary definitions. For example, in contrast to the respective conventional sets: BABY, CHILD, YOUNG, OLD and VERY OLD, the corresponding fuzzy sets allow any age in the interval [0, 120] years as a member of each of the above sets but with different memberships in [0, 1]. As a specific instance, the age 22 is a member of all the fuzzy sets but the membership of age (=22) to belong to the sets BABY, CHILD, YOUNG, OLD and VERY OLD respectively are 0.001, 0.01,1.00, 0.60 and 0.20. The above example makes sense in the line of reasoning

that an age of 22 corresponds to a young person, so the membership of age (=22) to be young is high (1.00). The relative grading of the other memberships thus can be easily understood from the usual meaning of the terms BABY, CHILD, OLD and VERY OLD. A fuzzy set thus can be formally defined as follows.

***Definition 2.1*:** A fuzzy set A is a set of ordered pairs, give

$$A = \{x, \quad \mu_A(x) : x \in X\} \tag{2.4}$$

where X is a universal set of objects (also called the universe of discourse) and $\mu_A(x)$ is the grade of membership of the object x in A. Usually, μ_A (x) lies in the closed interval of [0,1].

It may be added here that some authors [1] relax the range of membership from [0, 1] to [0, Rmax] where Rmax is a positive finite real number. One can easily convert [0, Rmax] to [0, 1] by dividing the membership values in the range [0, Rmax] by Rmax.

There are other notations of fuzzy sets as well. For instance, the ordered pair *(x, $\mu_A(x)$)* in the definition of fuzzy set is also written as $x/\mu_A(x)$ or μ_A/A as well. Let the elements of set X be $x_1, x_2, \ldots\ldots x_n$. Then the fuzzy set $A \subseteq X$ is denoted by any of the following nomenclature.

$$A = \{(x_1, \mu_A(x_1)), (x_2, \mu_A(x_2)), \ldots\ldots (x_n, \mu_A(x_n))\}$$

Or $$A = \{x_1 / \mu_A(x_1), x_2 / \mu_A(x_2), \ldots\ldots x_n / \mu_A(x_n)\}$$

Or $$A = \{x_1 / \mu_A(x_1) + x_2 / \mu_A(x_2) + \ldots\ldots + x_n / \mu_A(x_n)\}$$

Or $$A = \{\mu_A(x_1) / x_1 + \mu_A(x_2) / x_2 + \ldots\ldots + \mu_A(x_n) / x_n\}$$

Or $$A = \{\mu_A(x_1) / x_1, \mu_A(x_2) / x_2, \ldots\ldots \mu_A(x_n) / x_n\}$$

In this book we used the last option. The details of membership function $\mu_A(x)$ is formalized below.

2.2.1 Membership Functions

The grade of membership $\mu_A(x)$ maps the object or its attribute x to positive real numbers in the interval [0, 1]. Because of its mapping characteristics like a function, it is called membership function. A formal definition of the membership function is given below for the convenience of the readers.

Definition 2.2: A membership function $\mu_A(x)$ is characterized by the following mapping:

$$\mu_A : x \to [0,1], \qquad x \in X \tag{2.5}$$

where x is a real number describing an object or its attribute and X is the universe of discourse and A is a subset of X.

A question that naturally arises is: how to construct a membership function? The following examples provide a thorough insight to the selection of the membership functions.

Example 2.1: Consider the problem of defining BABY, CHILD, YOUNG, OLD and VERY OLD by membership functions. The closer the age of a person to 0, the higher is his/her membership to be a BABY. So, if x is the age of the person, we can define BABY as follows:

$$BABY = \{x, \mu_{BABY}(x)\} \quad where \quad \mu_{BABY}(x) = \exp(-x) \tag{2.6}$$

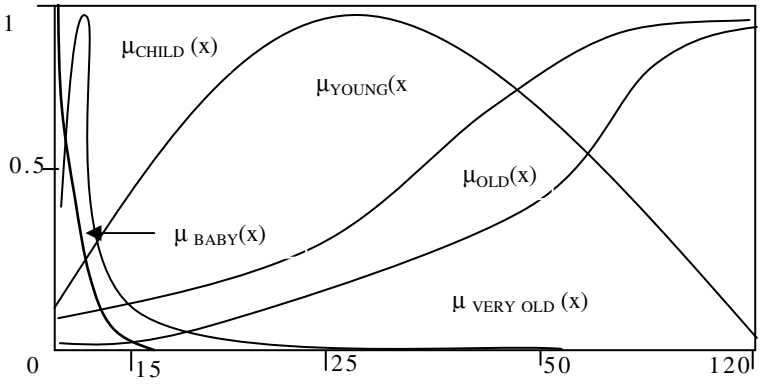

Fig. 2.1 Membership curves for the fuzzy sets: BABY, CHILD, YOUNG, OLD and VERY OLD. The x-axis denotes the age in years and the y-axis denotes the memberships of the given fuzzy sets at different ages.

Thus as $x \to 0$, $\mu_{BABY}(x) \to 1$. Further, as x increases, $\mu_{BABY}(x)$ decreases exponentially. The membership function $\mu_{BABY}(x)$ can also be designed to have a controlled decrease with increasing x by including a factor α to x in exp(-x). Thus,

$$\mu_{BABY}(x) = \exp(-\alpha x) \quad for\ \alpha > 0 \tag{2.7}$$

Larger the value of α, the higher is the falling rate of $\mu_{BABY}(x)$ over x. In a similar manner we can define the membership functions for CHILD, YOUNG, OLD and VERY OLD fuzzy sets. But before representing them mathematically let us take a look at them.

The membership curves for the fuzzy sets: BABY, CHILD, YOUNG, OLD and VERY OLD are shown in Fig. 2.1. The curve for CHILD fuzzy set has the peak at some age slightly greater than 0 and has a sharp fall off around the peak. The logical interpretation of this directly follows from the meaning of the word child.

The membership curve for the fuzzy set YOUNG has a peak at age x=25 and falls off very slowly on both sides around the peak.

As youth is the most charming period of the human beings, we prefer to call people YOUNG even if they are away from 25 on either side. If the readers' view is different they can allow a sharp falloff of the curve around the age x=25. One interesting point to note about the OLD and VERY OLD membership curves is that OLD curve throughout has a higher membership than the VERY OLD curve until both saturate at age x = 120 years. This is meaningful because if someone is called VERY OLD then he must be OLD, but the converse may not be true.

There are many ways to represent the membership functions shown in Fig. 2.1 by mathematical functions. One such representation is given below:

$$\mu_{\text{CHILD}}(x) = ax^2/(1 + bx^2 + xc), \qquad a, b, c > 0 \tag{2.8}$$

$$\mu_{\text{YOUNG}}(x) = \exp[-(x-25)^2/2\sigma^2], \qquad \sigma > 0 \tag{2.9}$$

$$\mu_{\text{OLD}}(x) = 1 - \exp(dx^2) \qquad d > 0 \tag{2.10}$$

$$\text{and } \mu_{\text{VERYOLD}}(x) = 1 - \exp(-dx), \qquad d > 0 \tag{2.11}$$

The parameters a, b, c and d in the above membership functions are selected intuitively by the experts based on their subjective judgement in the respective domains. Tuning of these parameters is needed to control the curvature and sharp changes on the curves around some selected x-values.

2.2.2 Continuous and Discrete Membership Functions

The universe of discourse (or simply the universe) of a fuzzy set may exist for both discrete and continuous spectrum. For example, the roll number of students in a class is a discrete universe. On the other hand the height of persons is a continuous universe as height may take up any values between 4' to 8'. It may be mentioned here that a continuous universe is sometimes sampled at regular or irregular intervals for using it as a discrete universe. The membership curve of YOUNG in Fig. 2.1 may be, for instance, discretized at age x= 18, 22, 24, 28. This is an example of non-uniform/ irregular sampling as the intervals of sampling 18-22, 22-24, 24-28 are unequal. The membership curve of YOUNG may alternatively be sampled at a regular interval of age x=18, 20, 22, 24, say. This is an example of uniform/ regular sampling. Fig. 2.2(a) and (b) describe the instances of the non-uniform and uniform sampling of the YOUNG membership curve.

2.2.3 Fuzzy Implication Rules

Reasoning informally refers to generation of inferences from a given set of facts and rules. The subject logic is concerned with the formalization of methodology

and principles of reasoning. Production rules are synonymously called implication rules in traditional logic. Implication rules of traditional logic are extended with fuzzy linguistic variables. In this section, the implication rules in the classical (propositional/ predicate) logic and their extension in fuzzy domain are introduced.

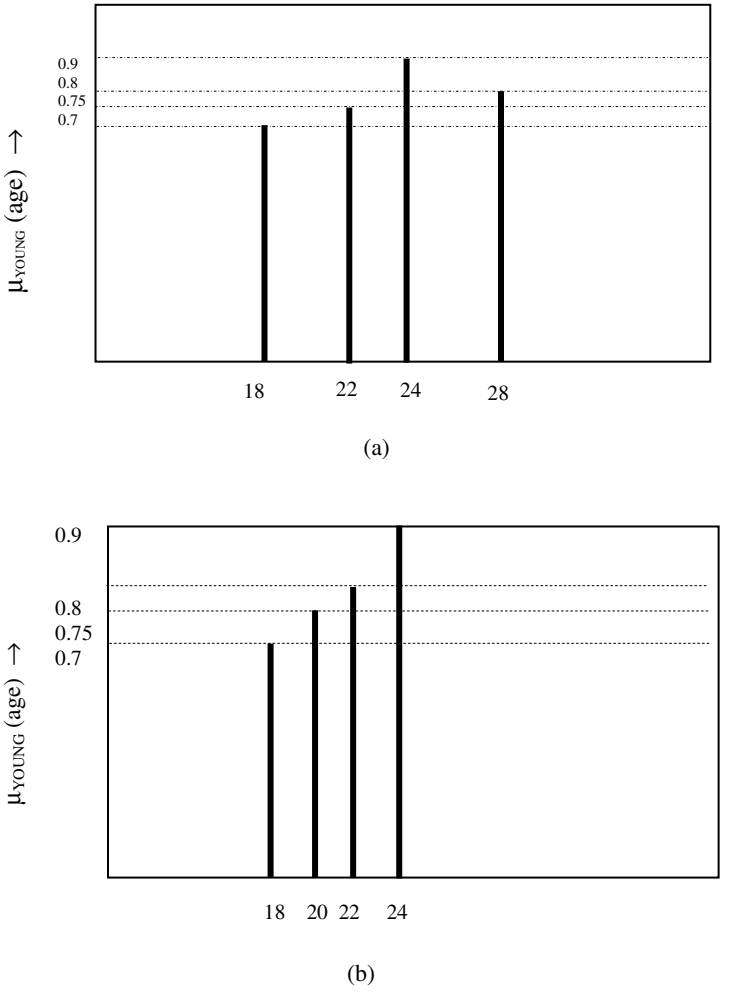

Fig. 2.2 (a) Non-uniform and (b) uniform sampling of the YOUNG membership curve.

Let us try to formalize the rule: IF x is a banana and x is yellow THEN x is ripe in predicate logic. For this formalization, we define 3 predicates Banana(x), Yellow(x) and Ripe(x), where each of these predicates has only 2 possible truth

values: true or false. Using a IF-THEN (implication) operator, the above rule can be written as:

$$Banana(x),\ Yellow(x) \rightarrow Ripe(x).$$

where the comma in the left side of the implication sign (→) denotes logical conjunction (AND) of the antecedent predicates.

In order to allow instantiation of the antecedent predicates with the contents of the WM, we consider 6 fuzzy sets: YELLOW, VERY-YELLOW, MORE-OR-LESS-YELLOW, RIPE, VERY-RIPE and MORE-OR-LESS-RIPE. Fuzzy extensions of the last rule then may be re-stated as follows:

Rule 1: IF a banana is YELLOW
THEN it is RIPE.
Rule 2: IF a banana is VERY-YELLOW
THEN it is VERY-RIPE.
Rule 3: IF a banana is MORE-OR-LESS-YELLOW
THEN it is MORE-OR-LESS-RIPE.

It may be added here that unlike production systems, the logic of fuzzy sets allows firing of these 3 rules concurrently in presence of a data element concerning color of a banana in the WM. Thus conflict resolution is not employed in fuzzy logic.

Formally, let x be a linguistic variable in a universe X, and A1, A2 and A3 are three fuzzy sets under the universe X. Also assume that y be another linguistic variable in a universe Y and B1 , B2 and B3 are 3 fuzzy sets under Y. Then the implication rules between variable x and y may be described as

Rule1: IF x is A1 THEN y is B1.
Rule2: IF x is A2 THEN y is B2.
Rule 3: IF x is A3 THEN y is B3.

Suppose the WM contains x is A', where A' is semantically close to A1, A2 and A3 respectively. In the subsequent sections, we demonstrate the evaluation procedure of y is B' from the known membership distributions of x is A_1, y is B_1, x is A_2, y is B_2, x is A_3, y is B_3 and x is A'

2.2.4 Fuzzy Implication Relations

A fuzzy implication relation [2] for a given rule: IF x is Ai THEN y is Bi is formally denoted by

$$R_i(x, y) = \left\{ \mu_{R_i}(x, y) / (x, y) \right\} \tag{2.12}$$

where the membership function $\mu_{R_i}(x, y)$ is constructed intuitively by many alternative ways. One typical implication relations is presented below.

Mamdani Implication: Mamdani proposed the following two implication functions [7, 12].

$$\mu_{R_i}(x, y) = Min[\mu_{A_i}(x), \mu_{B_i}(y)] \tag{2.13}$$

$$\text{OR} \quad \mu_{R_i}(x, y) = \mu_{A_i}(x\mu_{A_i}(x)\mu_{B_i}(y) \tag{2.14}$$

Mamdani implication functions are most widely used implications in fuzzy systems and fuzzy control engineering. This implication relation is constructed based on the assumption that fuzzy IF-THEN rules are local. For example, consider the implication rule: IF height is TALL THEN speed is HIGH. By Mamdani's implication function, we do not want to mean: IF height is SHORT THEN speed is SLOW. The second rule is rather an example of a non-local rule. The knowledge engineer thus has to decide whether he prefers local or non-local rules. If he prefers local rules then Mamdani's implication relation should be used.

2.2.5 Fuzzy Logic

The logic of fuzzy sets, also called fuzzy logic, is an extension of the classical propositional logic from 2 perspectives. First, instead of binary valuation space (truth/falsehood) of the propositional logic, fuzzy logic provides a multi-valued truth-space in [0, 1]. Secondly, propositional logic generates inferences based on the complete matching of the antecedent clauses with the available data, whereas fuzzy logic is capable of generating inferences even when a partial matching of the antecedent clauses against data elements in WM exists. In this section, we present 3 typical propositional inference rules and describe their possible extensions in fuzzy logic.

2.2.6 Typical Propositional Inference Rules

Let p, q and r be 3 propositions. The following 3 propositional inference rules are commonly used for logical inferencing.

- Modus Ponens: Given a proposition p and a propositional implication rule p→q, we can derive the inference q. Symbolically,

$$p \wedge (p \rightarrow q) \Rightarrow q \tag{2.15}$$

  where $\Rightarrow$ denotes a logical provability operator. This above inference rule is well known as modus ponens.
- Modus Tollens: Given a proposition ¬q, and the implication rule p→q, we can derive the inference ¬p. symbolically,

$$\neg q \wedge (p \rightarrow q) \Rightarrow \neg p \tag{2.16}$$

  The above inference rule is known as modus tollens.

- Hypothetical Syllogism: Given 2 implication rules p→q and q→r, then we can easily derive a implication p→r. Symbolically,

$$(p \rightarrow q) \wedge (q \rightarrow r) \Rightarrow p \rightarrow r \tag{2.17}$$

The above inference rule is popularly known as hypothetical syllogism or chain rule.

2.2.7 Fuzzy Extension of the Inference Rules

The logic of fuzzy set provides a general framework for the extension of the above 3 propositional inference rules. Fuzzy extension of modus ponens, modus tollens and hypothetical syllogisms are called generalized modus ponens, generalized modus tollens and generalized hypothetical syllogism respectively.

Generalized Modus Ponens (GMP): Consider a fuzzy production rule: IF x is A then y is B, and a fuzzy fact: x is A'. The GMP inference rule then infers y is B'. Here A, B, A' and B' are fuzzy sets such that A' is close to A, and B' is close to B. The inference rule also states that the closer the A/ to A, the closer the B' to B. Symbolically, the GMP can be stated as follows:

Given: *IF x is A THEN y is B.*
Given: *x is A'*

Inferred: *y is B'*

Generalized Modus Tollens (GMT): Given a fuzzy production rule: IF x is A THEN y is B, and a fuzzy fact y is B', the GMT then infers x is A', where the more is the difference between B' and B, the more is the difference between A' and A. Symbolically, the GMT is stated as follows:

Given: *IF x is A THEN y is B.*
Given: *y is B'*

Inferred: *x is A'.*

Generalized Hypothetical Syllogism (GHS): Given 2 fuzzy production rules: IF x is A THEN y is B, and IF y is B' THEN z is C', where A, B and C are 3 fuzzy sets, and B' is close to B. Then the GHS infers : If x is in A then z is in C'. The closer the B' to B, the closer the C' to C. Symbolically, we can state this rule as follows:

Given: *IF x is A THEN y is B.*
Given: *IF y is B' THEN z is C*

Inferred: If x is in A then z is in C'.

In the above definition of the inference rules, we just mentioned that $A^{/}$ is close to A, $B^{/}$ is close to B and the like. But we did not mention what we exactly mean by "close to". In fact "close to" can take any of the following forms: VERY, VERY-VERY, MORE-OR-LESS, NOT, ABOUT-TO, AROUND and other fuzzy hedges that mean a fuzzy set A/ are approximately similar to A.

2.2.8 The Compositional Rule of Inference

In this section we present the methodology for the evaluation of fuzzy inferences for GMP, GMT and GHS. This, however, calls for formalization of a fuzzy rule called the compositional rule of inference. The compositional rule of inference is usually applied to 2 fuzzy membership distribution, one of which usually have a smaller number of linguistic variables. The rule extends the latter membership distribution cylindrically, so as to increase its number of linguistic variables to the former distribution. The intersection of the former and the resulting distribution is then projected to desired axes. The whole process is referred to as the compositional rule of inference. How exactly the compositional rule of inference is applied to determine the fuzzy inferences in GMP, GMT and GHS is presented below.

2.2.9 Computing Fuzzy Inferences Using GMP

Given a fuzzy production rule: IF x is A THEN y is B, and a fact y is $B^{/}$, we by GMT infer x is $A^{/}$. In this section we present the principle of determining the membership distribution of x is A/, $\mu_A/(x)$, from the membership distribution of y is $B^{/}$, $\mu_{B^{/}}(y)$, and the membership $\mu_R(x, y)$ of the fuzzy relation between the antecedent and consequent part of the given rule. The computation involves cylindrical extension of $\mu_{B^{/}}(y)$ to $\mu_{B^{/}}$ *CYL (x, y),* then intersection of $\mu_{B^{/}}$ CYL (x, y) with $\mu_R(x, y)$ and finally projection of the resulting relation ox X-axis. Thus, following the same steps as in the case of GMP, it can easily be shown that

$$\mu_{A'}(x) = \max_{y \in Y} \left[t\{\mu_{B'}(y), \mu_R(x, y)\} \right]. \tag{2.18}$$

When $\mu_{B^{/}}(y)$ and $\mu_R(x, y)$ are discrete relations represented by a row vector and a matrix respectively, we can represent the above result by the following max-min composition operation:

$$\mu_{A'}(x) = \mu_{B'}(y) \circ \left[\mu_R(x, y)\right]^T \tag{2.19}$$

where T denotes the transposition operator over the given relation.

Example 2.2: This example illustrates the computation of GMP using max-min composition operator.

We take the same $\mu_R(x, y)$ as in Example 2.1, and the $\mu_B/(y)$ we obtained as the result in that example, and plan to determine $\mu_A/(x)$ by the compositional rule of inference. Thus,

$$\mu_{A'}(x) = \mu_{B'}(y) \circ [\mu_R(x, y)]^T = [0.8 \;\; 0.6 \;\; 0.9] \circ \begin{bmatrix} 0.8 & 0.6 & 0.5 \\ 0.6 & 0.5 & 0.9 \\ 0.7 & 0.6 & 0.5 \end{bmatrix}^T$$
$$= [0.8 \;\; 0.9 \;\; 0.7]$$

It may be noted that $\mu_A'(x)$ thus obtained is not same as that presumed in Example 2.2.

2.3 Neural Nets

This section provides an introduction to biological neural networks and their mathematical models called artificial neural networks. Neural networks have proved them successful in pattern recognition, system identification, function approximation and many others.

2.3.1 Biological Neural Networks

The human nervous system consists of small unicellular structures called neurons. Neurons thus are fundamental units or building blocks of a biological nervous system. A neuron comprises of 5 elements: dendrite, Dendron, cell body, synapse and axon (Fig. 2.3). The dendrites receive signals from other neurons, muscles or sensory organs and carry them to a relatively thick fiber called Dendron. The Dendron carry the signals to the cell body, which in turn generates a composite signal based on the strength of the signals received from the dendrites. The composite signal is transmitted to the synapse through the axon (Fig. 2.3).

The synapse is like a potential barrier that controls the flow of signal from the axon of one neuron to the dendrites of other neurons. The transmission of signals from one neuron to another at a synapse is a complex chemical process. Generally, a specific chemical called neuro-transmitter is released from the transmitting end of the synapse. The transmitters lower or raise the potential barrier of the synapse. On lowering the potential barrier, the signal from the transmitting end can easily reach the other end of the synapse. On raising the barrier, a signal loss takes place and only a small component of the signal can reach the other end of the synapse. Flow of signal from one neuron to the others thus can be controlled by the synapse. When the influence of the synapse tends to activate the post-synaptic neuron, the synapse is called excitatory. On the other hand, when the synapse prohibits the passage of signal flow to the post-synaptic neuron, the synapse is called inhibitory. The synaptic ending of an axon thus is of excitatory or inhibitory nature.

How the neurons participate in the learning process of the human beings remained a mystery till this date. However, it is evident that the process of learning has a correspondence with the type and amount of neuro-transmitters released at the pre-synaptic end of the neurons. Thus for similar sensory/control/

motor actions a neuron releases the same type and amount of neuro-transmitters. How exactly the neurons perform the above task is an interesting topic of research for the biologists and medical researchers.

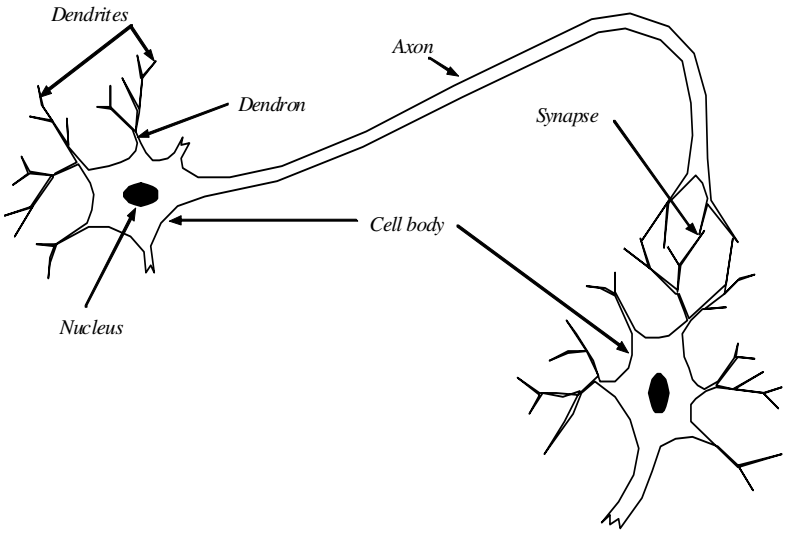

Fig. 2.3 A Biological Neural Net comprising of two neurons, where the dendrites of the second neuron receive signal from the synapse of the first neuron.

2.3.2 Artificial Neural Networks

Artificial neural networks are electrical analogue of the biological nervous system. A typical artificial neuron is mathematically represented by two modules: i) a linear activation/ inhibition module and ii) a non-linearity that limits the signal levels within a finite band. Fig. 2.4 presents the typical organization of an artificial neuron.

The summer in Fig. 2.4 takes the role of the cell body and the inputs of the summer may be treated like dendrites. The synapse is modeled by a non-linear function and the connection from the summer to the non-linear unit is like the axon. Here, Net is a linear combiner of the inputs x_1, x_2, …, x_n. Mathematically,

$$Net = \sum_{i=1}^{n} w_i x_i \tag{2.20}$$

where some of the inputs are excitatory (positive) and the rest are inhibitory (negative). 'Out' in the present context can take different mathematical forms. Some of the common forms are presented below.

$$Out = u(Net - th) \tag{2.21}$$

$$Out = Sgn(Net) \tag{2.22}$$

$$Out = 1/[1 + \exp(-Net)] \tag{2.23}$$

$$Out = \tanh(Net/2) \tag{2.24}$$

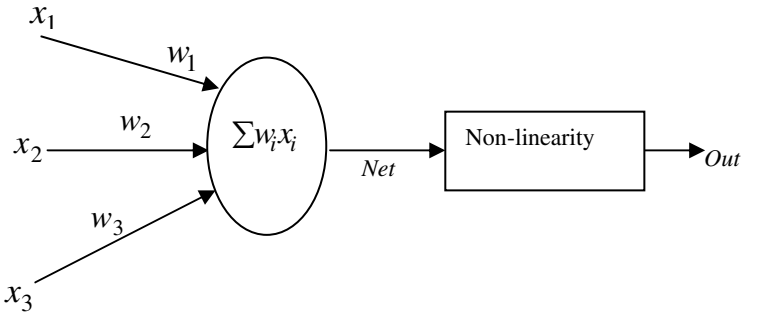

Fig. 2.4 A Typical Artificial Neuron

The mathematical form of Out can be smooth functions like sigmoid vide expression (2.23) or tanh vide expression (2.24) and sharp changing functions like step vide expression (2.21) or signum function (vide expression (2.22))

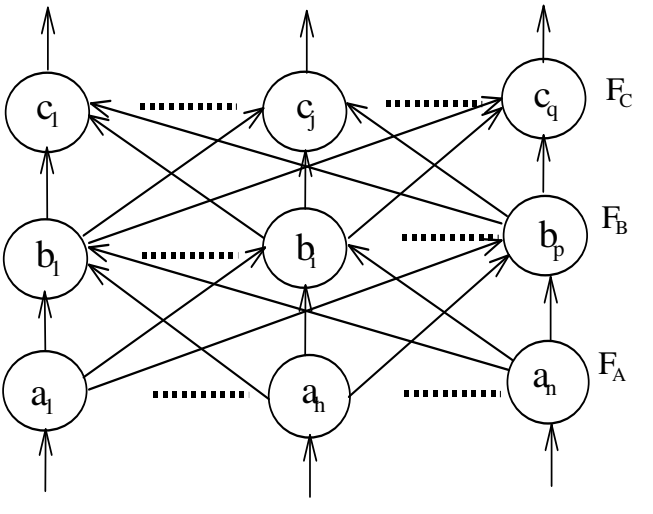

Fig. 2.5 A Feed-Forward Neural Net of 3 Layers

The unit step function u in expression (2.21) is formally defined as follows:

$$u(net - th) = \begin{cases} 1, & Net > th \\ 0, & otherwise \end{cases} \tag{2.25}$$

The signum function Sgn in expression (2.22) is formally defined as

$$Sgn(Net) = \begin{cases} +1, & Net > 0 \\ -1, & otherwise \end{cases} \tag{2.26}$$

The definitions of sigmoid and tanh are very standard and thus need no further explanation. Neurons in an artificial neural net are connected in different topological configurations. Two most common type of configurations are i)feed-forward and ii) feedback topology. Usually, a feed-forward network contains a number of layers, each layer consisting of a number of neurons (Fig. 2.4). Signal propagation in such networks usually take place in the forward direction only, i.e., signals from the i-th layer can be propagated to any layer following the i^{th} layer, for $i \geq 1$. In a recurrent neural network, there exists feedback from one or more neurons to others. Fig. 2.6 describes a recurrent network.

The most important aspect of an artificial neural net is its capability of learning. In the next section, we introduce the concepts of learning on artificial neural nets.

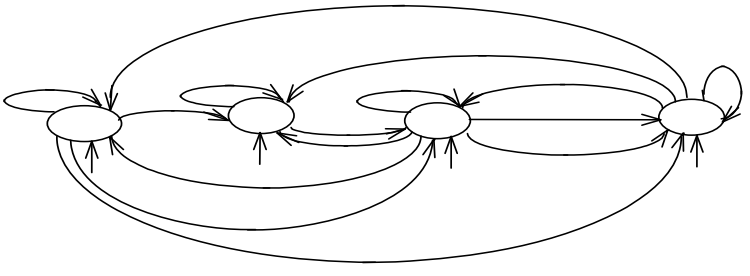

Fig. 2.6 A Typical Recurrent Neural Net

2.3.3 Principles of Learning in a Neural Net

Informally "encoding" or "learning" refers to adaptation of weights in a neural net. Thus until the weights converge to a steady state value, the process of encoding is continued. Adaptation of weights can be accomplished in a neural net by 4 different ways; they are supervised learning, unsupervised learning, reinforcement learning and competitive learning. A brief outline to the learning schemes is presented below.

- **Supervised Learning:** Supervised learning generally employs a trainer, who provides the input-output training instances of a given neural net. As an example, let us consider a pattern recognition problem, where we need to recognize an object from its feature- space. Here, the set of features such as size of the object and its shape described by its boundary descriptors, for instance, may be considered as input, while the type of the object such as books, pencils etc. may be treated as output of the neural net. Thus for n distinct objects, we require n-outputs of the neural net, each corresponding to one object.

When one of n-outputs has the maximum value, the object is regarded to fall within the particular class. Further for a large n, we can denote the output class by an encoded number, such as binary string. Thus for a given input feature vector, if a binary string 0011 appears at the output, we consider the object to belong to class 3.

Fig. 2.7 describes a scheme for supervised learning. Here, given a input vector I and a target vector T, we need to fix the weights in the network, such that T is produced at the output of the network when excited with the input I. How can we achieve this? First, we initialize the weights randomly. Then for the given input vector I, suppose the network generates the vector O at its output. An error vector E = T – O is then generated, and a supervised learning algorithm is used to adjust the Network parameters based on the error vector.

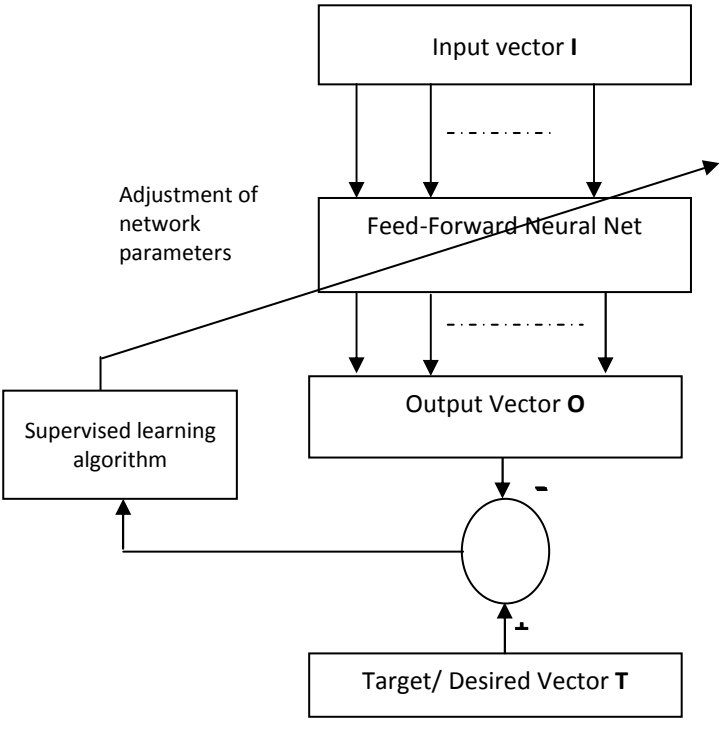

Fig. 2.7 A Simple Supervised Learning Scheme.

2.4 Swarm and Evolutionary Algorithms

Problems which involve global optimization over continuous spaces are ubiquitous throughout the scientific community. In general, the task is to optimize certain properties of a system by pertinently choosing the system parameters. For convenience, a system's parameters are usually represented as a vector.

The standard approach to an optimization problem begins by designing an objective function that can model the problem's objectives while incorporating any constraints. In a complex real life search problem, the search space may be a rough landscape, riddled with multiple local maxima/minima.

The objective function is very often non differentiable and/or discontinuous at a number of points. As for example consider the following functions shown in Fig. 2.8. Since the derivative based methods are of no help, other methods, combining mathematical analysis and random search came up for them. Imagine you scatter small robots in a Mountainous landscape. Those robots can follow the steepest path they found. When a robot reaches a peak, it claims that it has found the optimum. This method of hill climbing is very efficient, but there's no proof that the optimum has been found, each robot can be blocked in a local optimum. This type of method only works with reduced search spaces.

To tackle this kind of numerical optimization tasks over continuous search spaces, in 1995, two different algorithms were developed. First of them is the Particle Swarm Optimization (PSO) [3] while the second one goes by the name Differential Evolution (DE) [4]. Both of these algorithms do not require any gradient information of the function to be optimized, uses only primitive mathematical operators and is conceptually very simple. Unlike the conventional Genetic Algorithms (GA) [5] they can be implemented in any computer language very easily and requires minimal parameter tuning.

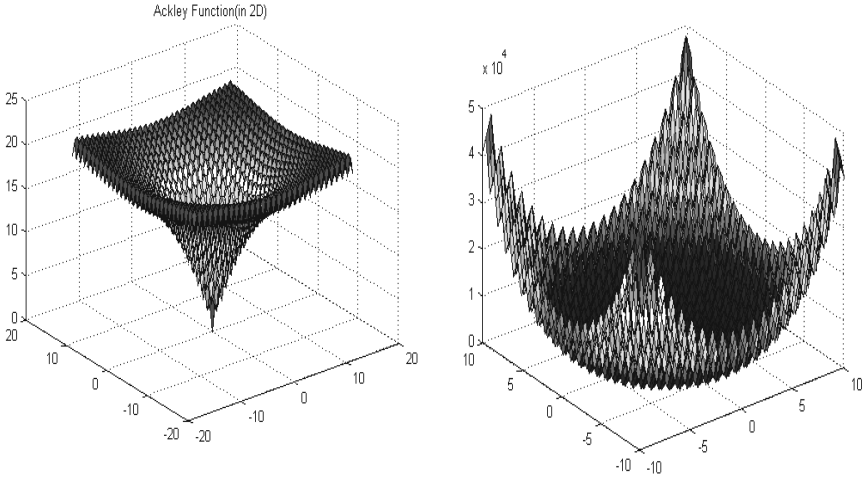

Fig. 2.8 Functions with Huge Number of Local Minima and Maxima

Their performance does not deteriorate severely with the growth of the search space dimensions as well. These issues perhaps have a great role in the popularity of the algorithms within the domain of machine intelligence and cybernetics.

2.4.1 Classical PSO

The concept of function-optimization by means of a particle swarm was introduced by James Kennedy and Russel C. Eberhart in an IEEE neural network conference paper from 1995 [3]. Suppose the global optimum of an n-dimensional function is to be located. The function may be mathematically represented as $f(x_1, x_2, x_3, ..., x_n) = f(\vec{X})$ where $\vec{X}$ is called the parameter vector which actually represents the set of independent variables. The task is to find out such a $\vec{X}$, that the function value $f(\vec{X})$ is either a minimum or a maximum denoted by f* in the search range. If the components of $\vec{X}$ assume real values then the task is to locate a particular point in the n dimensional hyperspace which is a continuum of such points.

Example 2.3 Consider the simplest two dimensional sphere function given by,

$$f(x_1, x_2) = f(\vec{X}) = x_1^2 + x_2^2$$

If x_1 and x_2 can assume real values only then by inspection it is pretty clear that the global minima of this function is at x_1=0, x_2=0 i.e. at the origin (0, 0) of the search space and the minimum value is f(0, 0) = f* = 0. No other point can be found in the x_1-x_2 plane at which value of the function is lower than f* = 0. Now the case of finding the optima is not so easy for functions like this one:

$$f(x_1, x_2) = x_1 \sin(4\pi x_2) - x_2 \sin(4\pi x_1 + \pi) + 1;$$

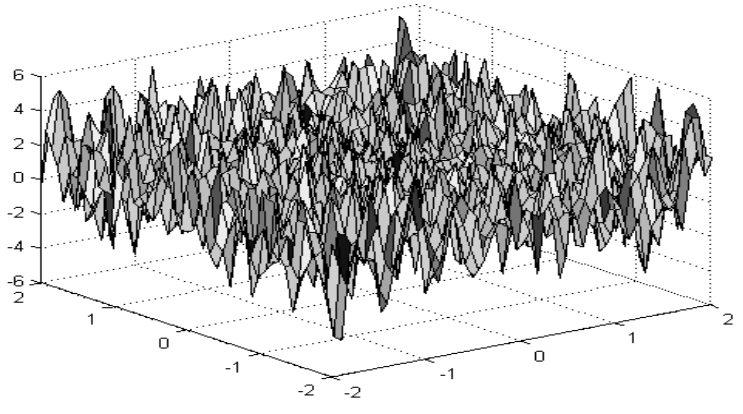

Fig. 2.9 Surface plot of the above-mentioned function

This function has multiple peaks and valleys and a rough fitness landscape. A surface plot of the function is shown in Fig. 2.9. To locate the global optima quickly on such a rough surface calls for parallel search techniques. Here many agents start from different initial locations and go on exploring the search space

until some (if not all) of the agents reach the global optimal position. The agents may communicate among themselves and share the fitness function values found by them.

PSO is in principle such a multi-agent parallel search technique. Particles are conceptual entities which fly through the multi-dimensional search space. At any particular instant each particle has a position and a velocity. The position vector of a particle with respect to the origin of the search space represents a trial solution of the each problem.

At the beginning a population of particles is initialized with random positions marked by vectors $\vec{X}_i$ and random velocities $\vec{V}_i$. Initial distribution of particles on a two dimensional search space may be illustrated in Fig. 2.10.

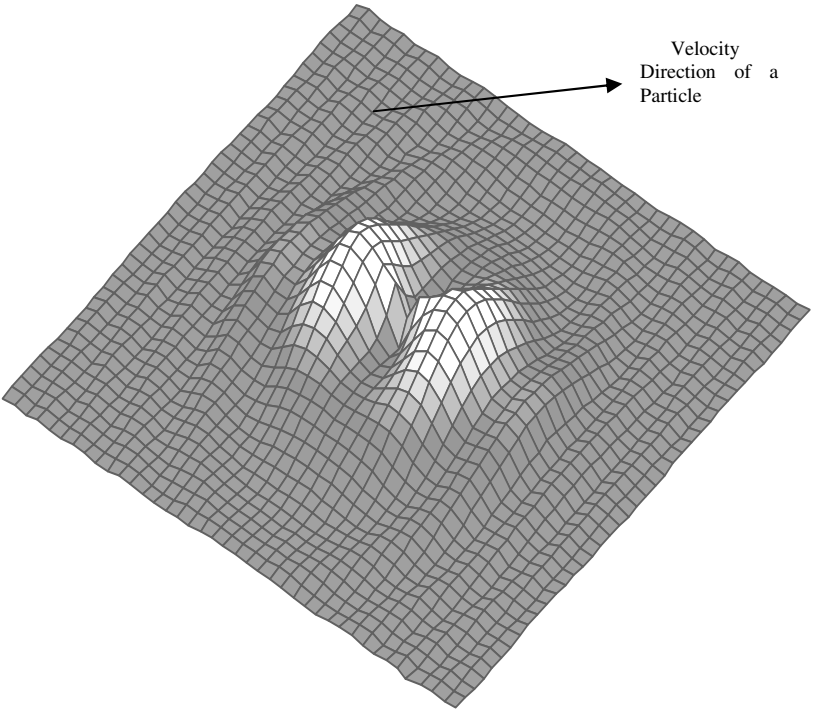

Fig. 2.10 Initial orientation of the swarm on a two dimensional fitness landscape

The population of such particles is called a 'swarm' *S*. A neighborhood relation *N* is defined in the swarm. *N* determines for any two particles P_i and P_j whether they are neighbors or not. Thus for any particle *P*, a neighborhood can be assigned as *N*(*P*), containing all the neighbors of that particle. Each particle *P* has two state variables:

- Its current position $\vec{x}(t)$
- Its current velocity $\vec{v}(t)$.

And also a small memory comprising,

- Its previous best position $\vec{p}(t)$ i.e. personal best experience.
- The best $\vec{p}(t)$ of all $P \in N(P)$: $\vec{g}(t)$ i.e. the best position found so far in the neighborhood of the particle.

The best $\vec{p}(t)$ of all $P \in N(P)$: $\vec{g}(t)$ i.e. the best position found so far in the neighborhood of the particle. The PSO scheme has the following algorithmic parameters:

- $V_{\max}$ or maximum velocity which restricts $\vec{V}_i(t)$ within the interval $[-v_{\max}, v_{mas}]$.
- An inertial weight factor ω [6].
- Two uniformly distributed random numbers φ_1 and φ_2 which respectively determine the influence of $\vec{p}(t)$ and $\vec{g}(t)$ on the velocity update formula.
- Two constant multiplier terms C_1 and C_2 known as "self confidence" and "swarm confidence" respectively.

Initially the settings for $\vec{p}(t)$ and $\vec{g}(t)$ are $\vec{p}(0) = \vec{g}(0) = \vec{x}(0)$ for all particles. Once the particles are initialized, the iterative optimization process begins where the positions and velocities of all the particles are altered by the following recursive equations. The equations are presented for the d^{th} dimension of the position and velocity of the i-th particle.

$$V_{id}(t+1) = \omega V_{id}(t) + C_1\varphi_1(P_d(t) - X_{id}(t)) + C_2\varphi_2(g_d(t) - X_{id}(t)) \tag{2.27}$$
$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1)$$

The first term in the velocity updating formula represents the inertial velocity of the particle. The second term $\vec{P}(t)$ involving represents the personal experience of each particle and is referred to as "cognitive part".

The last term of the same relation is interpreted as the "social term" which represents how an individual particle is influenced by the other members of its society. The velocity updating scheme has been presented in Fig. 2.11, using a humanoid agent in place of a particle on the spherical functional surface. After having calculated the velocities and position for the next time step *t*+1, the first iteration of the algorithm is completed.

Typically, this process is iterated for a certain number of time steps, or until some acceptable solution has been found by the algorithm or until an upper limit of CPU usage has been reached. Once the iterations are terminated, most of the

particles are expected to converge to a small radius surrounding the global optima of the search space. The ideal distribution of the particles after the algorithm is stopped has been shown in Fig. 2.12.

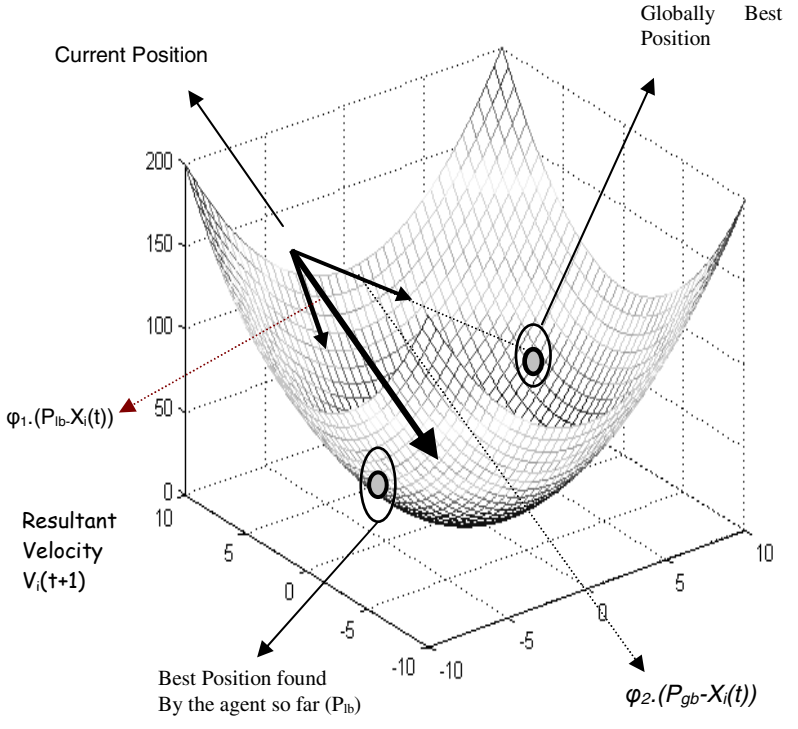

Fig. 2.11 Illustrating the velocity updating scheme of basic PSO

The algorithm can be summarized in the following pseudo code:

Procedure Particle_swarm_optimization
set t = 0;
Initialize φ_1, φ_2, V_{max} *and define N;*

While *(termination_condition = FALSE)*
{
$\forall p \in S$: *calculate* $\vec{v}(t+1)$ *and* $\vec{x}(t+1)$ *using equations (1);* $\forall p \in S$: *update* $\vec{p}(t+1)$ *with* $\vec{x}(t+1)$ *if* $f(\vec{x}(t+1))$ *is better than* $f(\vec{x}(t))$
$\forall p \in S$: *update* $\vec{g}(t+1)$ *with the best* $\vec{p}(t+1)$ *in N(p);*
}

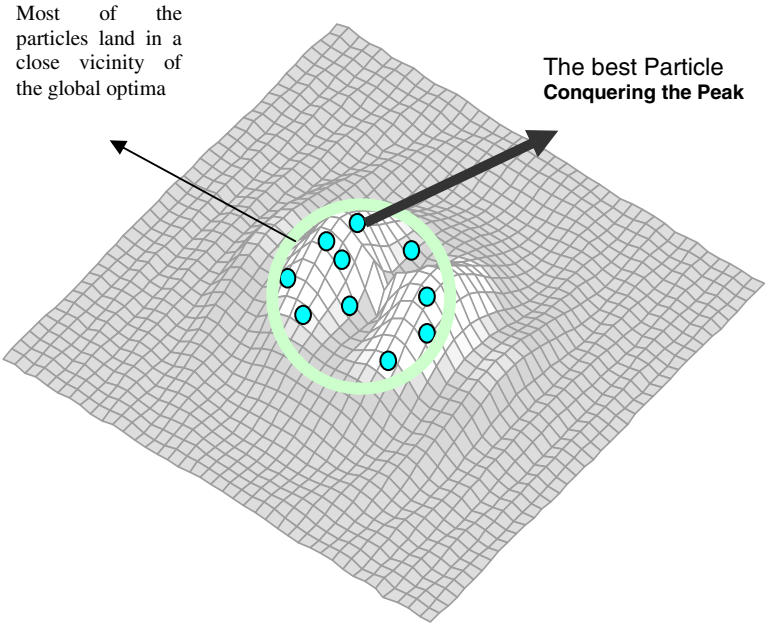

Fig. 2.12 Ideal distribution of the particles on a two dimensional fitness landscape after the algorithm is terminated

2.4.2 Differential Evolution

In 1995 Storn and Price made an attempt to replace the classical crossover and mutation operators in GA by alternative operators [4], and found a suitable vector differential operator to handle the problem. They proposed a new algorithm based on this operator, and called it Differential Evolution (DE). DE searches for a global optimum in a D-dimensional hyperspace. It begins with a randomly initialized population of D-dimensional real-valued parameter vectors. Each vector, also known as a 'genome' or 'chromosome', forms a candidate solution to the multi- dimensional optimization problem.

The initial population (at time t = 0) is chosen randomly and should be representative of as much of the search space as possible. Subsequent generations in DE can be represented by discrete time steps: t = 1, 2, ..., n etc. Since the parameter vectors are likely to be changed over different generations the following notation has been adopted here for representing the i-th vector of the population at the current generation (at time t):

$$\vec{X}_i(t) = [x_{i,1}(t), x_{i,2}(t), x_{i,3}(t).....x_{i,D}(t)]$$

For each parameter of the problem, there may be a certain range within which the value of the parameter must lie. At the beginning of a DE run, problem parameters or independent variables are initialized somewhere in their feasible numerical

range. So, if the j-th parameter of the given problem has its lower and upper bound as x_i^L and x_i^U respectively, then the j-th component of the i-th population member may be initialized as

$$x_{i,j}(0) = x_j^L + \text{rand}(0,1).(x_j^U - x_j^L) \tag{2.28}$$

where rand(0,1) is a uniformly distributed random number lying between 0 and 1.

For each individual vector $X_k(t)$ belonging to current population, DE randomly samples three other individuals $X_i(t)$, $X_j(t)$ and $X_m(t)$ from the same generation (for distinct *k, i, j* and *m*), calculates the difference of the components (chromosomes) of $X_i(t)$ and $X_j(t)$, scales it by a scalar R (ϵ [0,1]) and creates a trial offspring vector by adding the result to the chromosomes of $X_m(t)$. Thus for the n-th component of each parameter vector, we have

$$U_{k,n}(t+1) = \begin{cases} X_{m,n}(t) + R.(X_{i,n}(t) - X_{j,n}(t)) & \text{if } rand_n(0,1) < CR \\ X_{k,n}(t) & \text{otherwise} \end{cases} \tag{2.29}$$

where CR (ϵ[0,1]) is the crossover constant. This scheme is illustrated in Fig. 2.13.

To keep the population size constant over subsequent generations, the next step of the algorithm calls for 'selection' to determine which one between the parent and child will survive in the next generation (i.e. at time t+1). DE uses the Darwinian principle of "survival of the fittest" in its selection process which may be expressed as

$$\left.\begin{aligned} \overrightarrow{X_i}(t+1) = \overrightarrow{U_i}(t+1) \quad & \text{if} \quad f(\overrightarrow{U_i}(t+1)) < f(\overrightarrow{X_i}(t)) \\ = \overrightarrow{X_i}(t) \quad & \text{if} \quad f(\overrightarrow{U_i}(t+1)) > f(\overrightarrow{X_i}(t)) \end{aligned}\right\} \tag{2.30}$$

where f(.) is the function to be minimized. If the new offspring yields a better value of the fitness function, it replaces its parent in the next generation; otherwise the parent is retained in the population. Hence the population either gets better (with respect to the fitness values) or remains the same but never deteriorates.

The DE algorithm is outlined below:

Procedure Differential-evolution
Begin
Initialize population;
Evaluate fitness;
For i**=0 to max-iteration **do
Begin
Create Difference-Offspring;
Evaluate fitness;
***If** an offspring is better than its parent*
***Then** replace the parent by offspring in the next generation;*
End If;
End For;
End.

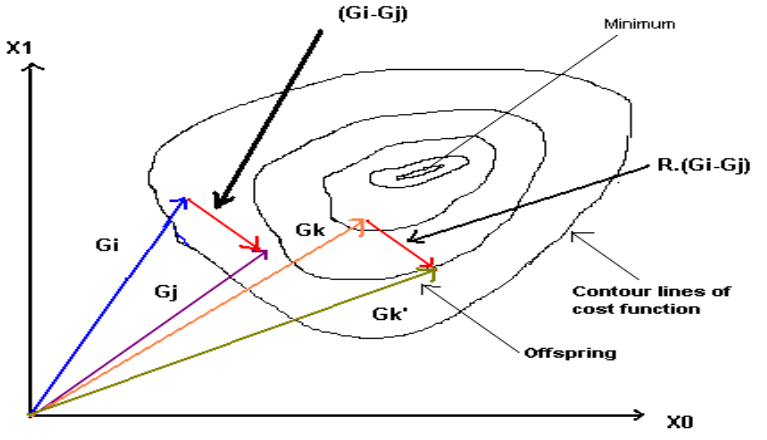

Fig. 2.13 Illustrating DE in 2-D parameter space

In the above algorithm, population is at first initialized to random values and fitness of each vector is judged according to some predefined cost function. The algorithm is then continued to generate population by invoking differential evolution and replacing parents by more fit offspring. The algorithm terminates when the fitness of the best genome is greater than a predefined value or maximum number of iterations has been attained.

2.4.2.1 Variants of Classical Differential Evolution

Generally in population-based search and optimization methods, considerably high diversity is necessary during the early part of the search to utilize the full range of the search space. On the other hand during the latter part of the search, when the algorithm is converging to the optimal solution, fine-tuning is important to locate the global optimum efficiently. Considering these issues, two new strategies [7] were proposed to improve the performance of the DE.

2.4.2.2 DERANDSF (DE with Random Scale Factor)

In the original DE [2] the difference vector $(\vec{X}_{r2}(t) - \vec{X}_{r3}(t))$ is scaled by a constant factor 'R'. The usual choice for this control parameter is a number between 0.4 and 1. We propose to vary this scale factor in a random manner in the range (0.5, 1) by using the relation

$$R = 0.5 * (1 + \text{rand}(0,1)) \tag{2.31}$$

where rand (0, 1) is a uniformly distributed random number within the range [0, 1]. We call this scheme DERANDSF (Differential Evolution with Random Scale Factor).

The mean value of the scale factor is 0.75. This allows for stochastic variations in the amplification of the difference vector and thus helps retain population diversity as the search progresses. Even when the tips of most of the population vectors point to locations clustered near a local optimum due to the randomly scaled difference vector, a new trial vector has fair chances of pointing at an even better location on the multimodal functional surface. Therefore the fitness of the best vector in a population is much less likely to get stagnant until a truly global optimum is reached.

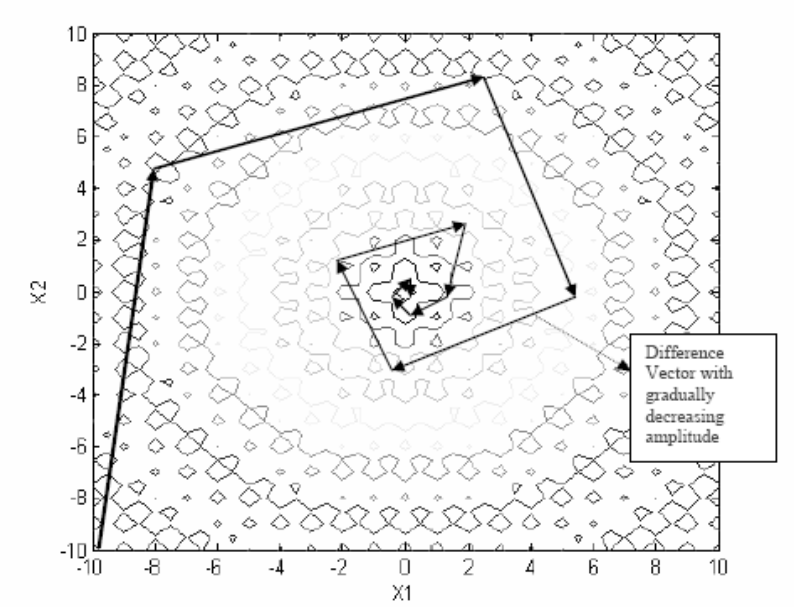

Fig. 2.14 Illustrating DETVSF scheme on two dimensional cost contours OF Ackley function

2.4.2.3 DETVSF (DE with Time Varying Scale Factor)

In most population-based optimization methods (except perhaps some hybrid global-local methods) it is generally believed to be a good idea to encourage the individuals (here, the tips of the trial vectors) to sample diverse zones of the search space during the early stages of the search. During the later stages it is important to adjust the movements of trial solutions finely so that they can explore the interior of a relatively small space in which the suspected global optimum lies. To meet this objective we reduce the value of the scale factor linearly with time from a (predetermined) maximum to a (predetermined) minimum value:

$$R = (R_{\max} - R_{\min}) * (MAXIT - iter) / MAXIT \tag{2.32}$$

where F_{max} and F_{min} are the maximum and minimum values of scale factor F, *iter* is the current iteration number and MAXIT is the maximum number of allowable iterations. The locus of the tip of the best vector in the population under this scheme may be illustrated as in Fig. 2.14.

2.5 Biogeography-Based Optimization (BBO)

Biogeography is the study of the distribution of biodiversity over space and time. It aims to analyze where organisms live, and in what abundance. Biogeography theory grew out of the work of Alfred Wallace [8] and Charles Darwin [9]. This gives rise to an interest in the distribution of organisms. The development of biogeography allowed scientists to test theories about the origin and dispersal of populations, which spurred its application in the field of the engineering. Just as what has happened in the past few years with the areas of computer intelligence [10, 11, 12], including genetic algorithms (GAs) [13, 14, 15], ant colony optimization (ACO) [16, 17, 18, 19], particle swarm optimization (PSO) [20, 21, 22], biogeography-based optimization (BBO) as a new type of evolutionary algorithm (EA) was recently proposed. This newest EA was introduced by Simon [23] in 2008 and demonstrated good optimization performance on various benchmark functions. In the original BBO paper, it was already proven that it is competitive with other famous EAs. If its highest potential is developed and applied to more practical problems, it could become a popular EA.

When a habitat is highly populated, it has many species and thus is likely to emigrate many species to nearby habitats, while few species immigrate into it, simply by virtue of the lack of room for immigrating species. In the same way, when a habitat is sparsely populated, it has few species and thus is likely to receive many immigrants, while only a few species emigrate because of their sparse populations.

The issue of whether or not those immigrants can survive after migration is another question, but the immigration of new species can raise the biological diversity of a habitat and thereby improve the habitat's suitability for other species. At least to this point, biogeography is a positive feedback phenomenon, and we regard this phenomenon of biogeography as an optimization process. This view of the environment as an optimizing system was suggested as early as 1990s [24]. In particular, some people maintain the view that "biogeography based on optimizing environmental conditions for biotic activity seems more appropriate than a definition based on homeostasis" [25]. In fact, there are many examples of the optimality of biogeographical processes to support this view, such as the Amazon rainforest [25] and the Krakatoa island phenomenon [26].

In another view, biogeography has often been considered as a process that enforces equilibrium in habitats. Over time, the countervailing forces of immigration and emigration result in an equilibrium level of species richness in a habitat with a large number of species. Namely, equilibrium can be seen as the point where the immigration and emigration curves intersect. The equilibrium viewpoint of biogeography was first popularized in the 1960s. Since then the equilibrium perspective has been increasingly questioned by scientists.

In a word, although the natural phenomenon of biogeographical as an optimization process has been challenged, adequate literature and ideas have been put forth to explain these challenges. It must be emphasized that optimality and equilibrium are only two different perspectives on the same phenomenon in biogeography, but this debate opens up many areas of further research for engineers.

As its name implies, BBO as a novel optimization method is based on the science of biogeography. The details of the BBO approach will be presented in the next section. Just as the mathematics of biology spurred the development of other biology-based optimization methods, we can incorporate certain behaviors of biogeography into BBO to improve its optimization performance. Some of these behaviors include the effect of geographical proximity on migration rates, nonlinear migration curves to better match nature, species populations, predator/prey relationships, the effect of varying species mobility on migration rates, directional momentum during migration, the effect of habitat area and isolation on migration rates, and many others.

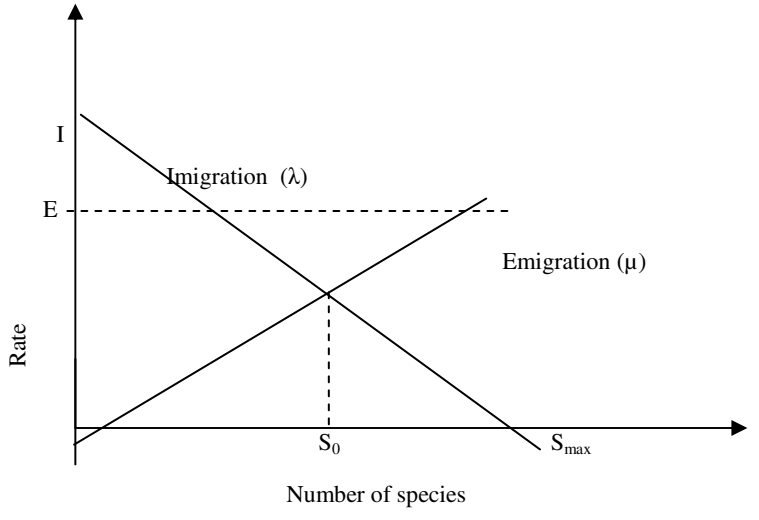

Fig. 2.15 Species model of a single habitat

The model of species abundance in a single habitat is shown in Fig. 2.15. The immigration rate λ and the emigration rate μ are functions of the number of species in the habitat. For the immigration curve, the maximum possible immigration rate to the habitat is I, which occurs when there are zero species in the habitat. As the number of species increases, the habitat becomes more crowded, fewer species are able to successfully survive immigration to the habitat, and the immigration rate decreases. The largest possible number of species that the habitat can support is S_{max}, at which point the immigration rate becomes zero.

For the emigration curve if there are no species in the habitat then the emigration rate must be zero. As the number of species increases, the habitat becomes more crowded; more species are able to leave the habitat to explore other possible residences, and the emigration rate increases. The maximum emigration rate is E, which occurs when the habitat contains the largest number of species that it can support.

The equilibrium number of species is S_o, at which point the immigration and emigration rates are equal. However, there may be occasional excursions from due to temporal effects. Positive excursions could be due to a sudden spurt of immigration (caused, perhaps, by an unusually large piece of flotsam arriving from a neighboring habitat), or a sudden burst of speciation (like a miniature Cambrian explosion). Negative excursions from could be due to disease, the introduction of an especially ravenous predator, or some other natural catastrophe. It can take a long time in nature for species counts to reach equilibrium after a major perturbation.

The immigration and emigration curves in shown in Fig. 2.16 as straight lines but, in general, they might be more complicated curves. Now, the probability Ps is the habitat contains exactly S species. Ps changes from time t to time (t + Δ t) as follows

$$P_s(t+\Delta t) = P_s(t)\,(1-\lambda_s \Delta t - \mu_s \Delta t) + P_{s-1}\lambda_{s-1}\Delta t + P_{s+1}\,\mu_{s+1}\,\Delta t \tag{2.33}$$

where λ_s and μ_s are the immigration and emigration rates when there are S species in the habitat. This equation holds because in order to have S species at $(t+\Delta t)$ time , one of the following conditions must hold:

- There were S species at time t, and no immigration or emigration occurred between t and $(t+\Delta t)$;
- There were *(S - 1)* species at time t, and one species immigrated;
- There were *(S + 1)* species at time , and one species emigrated.

It is assumed that Δ t is small enough so that the probability of more than one immigration or emigration can be ignored.

Taking the limit of (1) as Δ t → 0 gives equation (2) shown as follows:

$$\hat{P}_s = \begin{cases} -(\lambda_s+\mu_s)P_s + \mu_{s+1}P_{s+1} & S=0 \\ -(\lambda_s+\mu_s)P_s + \lambda_{s-1}P_{s-1} + \mu_{s+1}P_{s+1} & 1 \le S \le S_{\max}-1 \\ -(\lambda_s+\mu_s)P_s + \lambda_{s-1}P_{s-1} & S = S_{\max} \end{cases} \tag{2.34}$$

Say, $n = S_{\max}$ and $P = [P_0 P_1 P_2 .....P_n]^T$

Now, we can arrange the equations of equation (2) into the single matrix equation

$$\overset{o}{P} = AP \tag{2.35}$$

where the matrix A is given in the following equation:

$$A=\begin{bmatrix} -(\lambda_0+\mu_0) & \mu_1 & 0 & \ldots & 0 \\ \lambda_0 & -(\lambda_1+\mu_1) & \mu_2 & \ldots. & \ldots \\ \ldots & \ldots. & \ldots. & \ldots & \ldots \\ \ldots. & \ldots. & \lambda_{n-2} & -(\lambda_{n-1}+\mu_{n-1}) & \mu_n \\ 0 & \ldots & 0 & \lambda_{n-1} & -(\lambda_n+\mu_n) \end{bmatrix} \tag{2.36}$$

For the straight-line curves shown in Fig. 1, we have

$$\left.\begin{aligned} \mu_k &= \frac{Ek}{n} \\ \lambda_k &= I\left(1-\frac{k}{n}\right) \end{aligned}\right\} \tag{2.37}$$

Now for special case E= I , then

$$\lambda_k + \mu_k = E \tag{2.38}$$

According to the simplified form stated in equation (6), the species model will be the following type.

2.5.1 Migration

Suppose that we have a problem and a population of candidate solutions that can be represented as vectors of integers. Each integer in the solution vector is considered to be an SIV. The assessment for the goodness of the solutions has to be done. The solutions that are good are considered to be habitats with a high Habitat Suitability Index (HIS), and those that are poor are considered to be habitats with a low HSI. HSI is analogous to "fitness" in other population-based optimization algorithms (GAs, for example).

High HSI solutions represent habitats with many species, and low HSI solutions represent habitats with few species. The identical species curve (E = I) is considered for simplicity but the S value represented by the solution depends on its HSI. S_1 in Fig. 2.16 represents a low HSI solution, while S_2 represents a high HSI solution. S_1 in Fig.2.16 represents a habitat with only a few species, while S_2 represents a habitat with many species.

The immigration rate λ_1 for S_1 will be higher than the immigration rate λ_2 for S_2. The emigration rate $\mu 1$ for S_1 will be lower than the emigration rate $\mu 2$ for S_2.

The emigration and immigration rates of each solution probabilistically share information between habitats. With probability P_{mod}, each solution is modified based on other solutions. If a given solution is selected to be modified, then the immigration rate λ to probabilistically decide whether or not to modify each suitability index variable (SIV) in that solution.

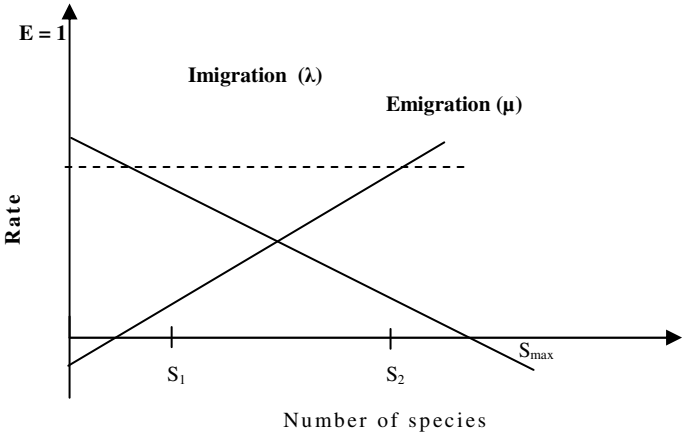

Fig. 2.16 S_1 is relatively a poor solution and S_2 relatively a good solution

If a given SIV in a given solution S_i selected to be modified, then the emigration rates μ of the other solutions to probabilistically decide which of the solutions should migrate a randomly selected SIV to solution S_i.

The BBO migration strategy is similar to the global recombination approach of the breeder GA and evolutionary strategies in which many parents can contribute to a single offspring, but it differs in at least one important aspect. In evolutionary strategies, global recombination is used to create new solutions, while BBO migration is used to change existing solutions. Global recombination in evolutionary strategy is a reproductive process, while migration in BBO is an adaptive process; it is used to modify existing islands.

To retain the best solutions in the population, some sort of elitism is incorporated. This prevents the best solutions from being corrupted by immigration.

2.5.2 Migration Algorithm

Habitat modification can loosely be described as follows:
Select H_i with probability proportional to λ_i
If H_i is selected
For j=1 to n
Select H_j with probability proportional to μ_j
If H_j is selected
Randomly select an SIV from H_j
Replace a random SIV in with
end
end
end

2.5.3 Mutation

A habitat's HSI can change suddenly due to apparently random events (unusually large flotsam arriving from a neighboring habitat, disease, natural catastrophes, etc.) The model of BBO as SIV mutation, and species count probabilities is used to determine mutation rates.

The probabilities of each species count will be governed by the differential equation given in 2.39. By looking at the equilibrium point on the species curve of Fig. 2.16, it is observed that low species counts and high species counts both have relatively low probabilities and medium species counts have high probabilities because they are near the equilibrium point.

Each population member has an associated probability, which indicates the likelihood that it was expected *a priori* to exist as a solution to the given problem. Very high HSI solutions and very low HSI solutions are equally improbable. Medium HIS solutions are relatively probable. If a given solution S has a low probability P_s , then it is surprising that it exists as a solution. It is, therefore, likely to mutate to some other solution. Conversely, a solution with a high probability is less likely to mutate to a different solution. The mutation rate that is inversely proportional to the solution probability,

$$m_i = m_{\max}\left(1 - \frac{P_i}{P_{\max}}\right) \tag{2.39}$$

where,

m_{max} is a user-defined parameter,

and $P_{max} = argmax\ P_i$, $i = 1,...NP$.

This mutation scheme tends to increase diversity among the population. Without this modification, the highly probable solutions will tend to be more dominant in the population. This mutation approach makes low HSI solutions likely to mutate, which gives them a chance of improving. It also makes high HSI solutions likely to mutate, which gives them a chance of improving even more than they already have. Note that we use an elitism approach to save the features of the habitat that has the best solution in the BBO process, so even if mutation ruins its HSI, we have saved it and can revert back to it if needed. So, we use mutation (a high risk process) on both poor solutions and good solutions. Those solutions that are average are hopefully improving already, and so we avoid mutating them (although there is still some mutation probability, except for the most probable solution).

Mutation Algorithm: *Mutation can be described as follows:*

For j=1 to m

Use λ_i *and* μ_i *to compute the probability* P_i

Select SIV $H_i(j)$ *with probability proportional to* P_i

If $H_i(j)$ *is selected*

Replace $H_i(j)$ with a randomly generated SIV
END
END

2.6 Summary

The chapter introduced fundamental techniques of computational intelligence with special reference to fuzzy sets, neuro-computing and evolutionary algorithms. Special emphasis has been given to swarm and evolutionary algorithms, in particular Biogeography Based Optimization, Particle Swarm Optimization and Differential Evolution algorithms. A brief overview is given to neural learning, particularly supervised learning. It also includes an overview on fuzzy reasoning, starting from the first principles.

References

1. Zimmermann, H.J.: Fuzzy Set Theory and Its Applications. Kluwer Academic, Dordrecht (1991)
2. Dubois, D., Prade, H.: Fuzzy Sets and Systems: Theory and Applications. Academic Press, NY (1980)
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
4. Storn, R., Price, K.: Differential evolution – A Simple and Efficient Heuristic for Global continuous spaces. Journal of Global Optimization 11(4), 341–359 (1997)
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
6. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Academic Press (2001) ISBN 1-55860-595-9
7. Das, S., Konar, A., Chakraborty, U.K.: Two Improved Differential Evolution Schemes for Faster Global Search. In: ACM-SIGEVO Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2005), Washington DC (June 2005)
8. Wallace, A.: The Geographical Distribution of Animals (Two Volumes). Adamant Media Corporation, Boston (2005)
9. Darwin, C.: The Origin of Species. Gramercy, New York (1995)
10. Hanski, I., Gilpin, M.: Metapopulation Biology. Academic, New York (1997)
11. Wesche, T., Goertler, G., Hubert, W.: Modified habitat suitabilityindex model for brown trout in southeastern Wyoming. North Amer. J. Fisheries Manage. 7, 232–237 (1987)
12. Hastings, A., Higgins, K.: Persistence of transients in spatially structured models. Science 263, 1133–1136 (1994)
13. Muhlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. Evol. Comput. 1, 25–49 (1993)
14. Back, T.: Evolutionary Algorithms in Theory and Practice. Oxford Univ. Press, Oxford (1996)
15. Parker, K., Melcher, K.: The modular aero-propulsion systems simulation (MAPSS) users' guide. NASA, Tech. Memo. 2004-212968 (2004)

16. Simon, D., Simon, D.L.: Kalman filter constraint switching for turbofan engine health estimation. Eur. J. Control 12, 331–343 (2006)
17. Simon, D.: Optimal State Estimation. Wiley, New York (2006)
18. Mushini, R., Simon, D.: On optimization of sensor selection for aircraft gas turbine engines. In: Proc. Int. Conf. Syst. Eng., Las Vegas, NV, pp. 9–14 (August 2005)
19. Chuan-Chong, C., Khee-Meng, K.: Principles and Techniques in Combinatorics. World Scientific, Singapore (1992)
20. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
21. Dorigo, M., Gambardella, L., Middendorf, M., Stutzle, T.: Special section on 'ant colony optimization'. IEEE Trans. Evol. Comput. 6(4), 317–365 (2002)
22. Blum, C.: Ant colony optimization: Introduction and recent trends. Phys. Life Reviews 2, 353–373 (2005)
23. Onwubolu, G., Babu, B.: New Optimization Techniques in Engineering. Springer, Berlin (2004)
24. Price, K., Storn, R.: Differential evolution. Dr. Dobb's Journal 22, 18–20, 22, 24, 78 (1997)
25. Storn, R.: System design by constraint adaptation and differential evolution. IEEE Trans. Evol. Comput. 3, 22–34 (1999)
26. Michalewicz, Z.: Genetic Algorithms Data Structures _ Evolution Programs. Springer, New York (1992)
27. Rumelhart, D.E., Zipser, D.: Feature discovery by competitive learning. Cognitive Science 9, 75–112 (1985)
28. Sejnowski, T.J.: Strong covariance with nonlinearly interacting neurons. J. Math Biology 4, 303–321 (1977)
29. Takeuchi, A., Amari, S.-I.: Formation of topographic maps and columnar microstructures. Biological Cybernetics 35, 63–74 (1979)
30. Yegnanarayana, B.: Artificial Neural Networks. Prentice-Hall of India, New Delhi (1988)
31. Baird, L.C., Moore, A.W.: Gradient descent for general reinforcement learning. In: Advances in Neural Information Processing Systems, vol. 11. The MIT Press (1999)
32. Bertsekas, D.P.: Dynamic Programming And Optimal Control, vol. 1 & 2. Athena Scientific, Belmont (1995b)
33. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8, 229–256 (1992)
34. Kullback, S.: Information theory and statistics. John Wiley and Sons, NY (1959)
35. Meuleau, N., Dorigo, M.: Ant colony optimization and stochastic gradient descent. Artificial Life 8(2), 103–121 (2002)