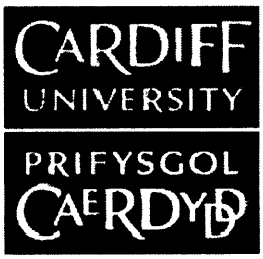


Enhanced Bees Algorithm with Fuzzy Logic and Kalman Filtering

A thesis submitted to
Cardiff University,
for the degree of

Doctor of Philosophy

by

Ahmed Haj Darwish

Intelligent Systems Research Laboratory

Manufacturing Engineering Centre

Cardiff University

United Kingdom

2009

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

The Bees Algorithm is a new population-based optimisation procedure which employs a combination of global exploratory and local exploitatory search.

This thesis introduces an enhanced version of the Bees Algorithm which implements a fuzzy logic system for greedy selection of local search sites. The proposed fuzzy greedy selection system reduces the number of parameters needed to run the Bees Algorithm. The proposed algorithm has been applied to a number of benchmark function optimisation problems to demonstrate its robustness and self-organising ability.

The Bees Algorithm in both its basic and enhanced forms has been used to optimise the parameters of a fuzzy logic controller. The purpose of the controller is to stabilise and balance an under-actuated two-link acrobatic robot (ACROBOT) in the upright position.

Kalman filtering, as a fast convergence gradient-based optimisation method, is introduced as an alternative to random neighbourhood search to guide worker bees speedily towards the optima of local search sites. The proposed method has been used to tune membership functions for a fuzzy logic system.

Finally, the fuzzy greedy selection system is enhanced by using multiple independent criteria to select local search sites. The enhanced fuzzy selection

system has again been used with Kalman filtering to speed up the Bees Algorithm. The resulting algorithm has been applied to train a Radial Basis Function (RBF) neural network for wood defect identification.

The results obtained show that the changes made to the Bees Algorithm in this research have significantly improved its performance. This is because these enhancements maintain the robust global search attribute of the Bees Algorithm and improve its local search procedure.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. D.T. Pham for his excellent supervision, continuous encouragement and support. He was a brilliant supervisor.

I also want to thank all my colleagues at the Intelligent Systems Research Laboratory, who were very good to me and very helpful whenever I needed them.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI	**Artificial Intelligence**
ABC	**Artificial Bees Colony**
ACO	**Ant Colony Optimisation**
ACROBOT	**ACRObatic roBOT**
ANN	**Artificial Neural Network**
ANTS	**Ant Colony System**
AVI	**Automated Visual Inspection**
BA	**The Bees Algorithm**
EKF	**Extended Kalman Filter**
ES	**Expert System**
FL	**Fuzzy Logic**
FLS	**Fuzzy Logic System**
GA	**Genetic Algorithm**
GMP	**Generalised Modus Ponens**
GMT	**Generalised Modus Tollens**
LQR	**Linear Quadratic Regulator**
LVQ	**Learning Vector Quantisation**
MDC	**Minimum Distance Classifier**
MLP	**Multi-Layered Perceptrons**

NE-SIMPSA	**Stochastic Simulated Annealing Optimisation Procedure**
PCB	**Printed Circuit Board**
PID	**Proportional–Integral–Derivative**
PSO	**Particle Swarm Optimisation**
RBF	**Radial Basis Function**
SA	**Simulated Annealing**
SIMPSA	**Deterministic Simplex method**
SVM	**Support Vector Machine**
TS	**Tabu Search**
TSP	**Travel Salesman Problem**
VBA	**Virtual Bees Algorithm**

CHAPTER 1. INTRODUCTION

1.1. Background

The rapid development of engineering sciences and increases in the number of complex processes in industry and manufacturing mean that traditional optimisation techniques are no longer adequate to solve complex multi-variable optimisation problems with large numbers of parameters. These usually require intelligent optimisation tools such as the Bees Algorithm (Pham et al. 2005; Pham et al. 2006b).

Studies in artificial intelligence have resulted in a number of intelligent optimisation algorithms. Many of them are inspired by biological phenomena like the natural foraging behaviour of honey bees, as in the case of the Bees Algorithm. This algorithm is a new population-based optimisation procedure which employs a combination of global exploratory and local exploitatory search. The algorithm requires a large number of parameters to be correctly set before it can be run. This work introduces a number of enhancements to the Bees Algorithm to reduce the efforts needed to produce the best results. A proposed enhancement concerns reducing the number of parameters needed to run the algorithm. Another improvement relates to the speeding-up of the search process.

1.2. Research Aim and Objectives

The overall aim of this work was to prove the hypothesis that (i) fuzzy logic can be adopted to reduce the number of parameters of the Bees Algorithm and (ii) a gradient-based prediction tool such as the Kalman filter will enhance the speed of the algorithm.

The following objectives were set to achieve this aim.

- Survey current intelligent optimisation algorithms, including the Bees Algorithm.
- Develop new forms of the Bees Algorithm to accelerate the search process and to reduce the number of parameters needed to run the Bees Algorithm.
- Apply the proposed optimisation tools to different categories of continuous optimisation problems.
- Validate the different versions of the proposed algorithm by applying them to different benchmark optimisation problems and compare the results obtained with those of other optimisation methods.

To achieve the above objectives, the following methodology was adopted:

- Review of previous work: an extensive survey was performed of the state of the art in intelligent optimisation techniques, focusing on bees-inspired algorithms, to identify research trends and potential solutions.
- Algorithm development and evaluation: the standard Bees Algorithm was extended by adding a fuzzy logic system for greedy selection of local search sites and a Kalman filter for neighbourhood search. The performance of the new versions of the algorithm was evaluated by computer simulation to solve a number of benchmark problems. The results obtained were compared with those of other optimisation techniques to assess the effectiveness of the proposed methods.

1.3. Thesis Organisation

The remainder of the thesis is organised as follows:

Chapter 2 reviews the background literature on intelligent optimisation algorithms relevant to the work presented in the thesis. This covers material on Expert Systems, Tabu Search, Simulated Annealing, Artificial Neural Networks, Kalman filtering, Fuzzy Logic, the Genetic Algorithm, Ant Colony Optimisation, Particle Swarm Optimisation and bees-inspired algorithms including the Bees Algorithm.

Chapter 3 describes an enhanced form of the Bees Algorithm and the application of fuzzy logic to the algorithm to reduce the number of parameters needed to run

it. The new algorithm employs a fuzzy greedy system to select local search sites. The chapter gives the results obtained in applying the algorithm to standard test problems.

Chapter 4 presents the application of the enhanced Bees Algorithm to the problem of optimising a fuzzy logic controller for an under-actuated two-link acrobatic robot. The results obtained demonstrate the superior performance of the new algorithm compared to the basic version.

Chapter 5 discusses another development of the Bees Algorithm using Kalman filtering and its application to optimise membership functions for a fuzzy logic system. This chapter also presents an enhanced fuzzy selection system and describes its application to the Bees Algorithm with Kalman filtering to train a Radial Basis Function (RBF) neural network for wood defect identification.

Chapter 6 concludes the thesis and suggests areas for further investigation.

CHAPTER 2. INTELLIGENT OPTIMISATION ALGORITHMS

2.1. Preliminaries

A recent trend in the science of Artificial Intelligence (AI) is the utilisation of tools to solve optimisation problems which are defined as minimisation of loss functions (Spall 2003). AI may be defined as computer procedures that simulate the human mind and the natural behaviour of living creatures to model and solve complex ill-defined problems (Tsoukalas and Uhrig 1997).

Such problem solvers are called intelligent optimisation algorithms, which include a number of techniques such as Expert Systems (ES) (Negnevitsky 2005), Artificial Neural Networks (ANN) (Haykin 1999), Fuzzy Logic Systems (FLS) (Tanaka 1997; Yen and Langari 1999), the Genetic Algorithm (GA) (Goldberg 1989; Holland 1975, 1992) and recently swarm-based algorithms including Ant Colony Optimisation (ACO) (Dorigo and Blum 2005), Particle Swarm Optimisation (PSO) (Kennedy and Eberhart 1995) and the Bees Algorithm (BA) (Pham et al. 2006b). These algorithms have been also used to solve a large number of complex problems (Pham et al. 2008e) and to plan

collaboratively arrangements for multi-process systems (Awadalla 2005; Pham et al. 2007b).

Moreover, a number of heuristic methods such as Tabu Search (TS) (Glover 1989, 1990) and Simulated Annealing (SA) (Kirkpatrick et al. 1983) have been used to solve optimisation problems and are classified as intelligent optimisation techniques (Pham and Karaboga 1999). The Kalman filter (Russell and Norvig 2004), which is a recursive estimator, has also been applied to solve a number of optimisation problems.

2.2. Expert Systems

An Expert System (ES) is a means of extracting and summarising human experience in a rule base, and this stored knowledge is then used to solve problems in a similar manner to the human brain (Efraim and Louis 1992). An ES employs formal logic in forward or backward chaining reasoning to determine required actions depending on measured or acquired inputs. It may exceed human performance (Weiss and Kulikowski 1991). ESs have been used to solve a number of combinatorial problems such as planning and scheduling problems (Metaxiotis et al. 2002) and Printed Circuit Board (PCB) assembly planning problems (Sanii and Liau 1993; Shih et al. 1996).

2.3. Tabu Search

Tabu Search (TS) was introduced by (Glover 1989, 1990) to solve combinatorial problems. TS is considered an iterative search algorithm assigned with a flexible memory (Pham and Karaboga 1999). It has the capability to purge local optima and to find a global optimum for a multimodal combinatorial problem. A tabu list is implemented to determine which solutions may be obtained by a move from the current solution; however that does not mean it always obtains better solutions and it may return to recently visited solutions. Three strategies have been implemented in TS (Pham and Karaboga 1999). The first one is the forbidding strategy that manages the moves to enter the tabu list; the second strategy is the freeing strategy which regulates the moves to exit the tabu list; and finally the short term strategy which organises the relations between the forbidding and freeing strategies. Figure 2.1 shows a flow chart of the standard TS algorithm.

TS has been applied to a variety of applications such as neural networks training (Ye et al. 2007), fuzzy logic applications (Bagis 2003), the Travelling Salesman Problem (TSP) (Fang et al. 2003; Voudouris and Tsang 1999), vehicle routing problems (Gendreau et al. 1996) and PCB assembly planning (Saad and Khalil 2005).

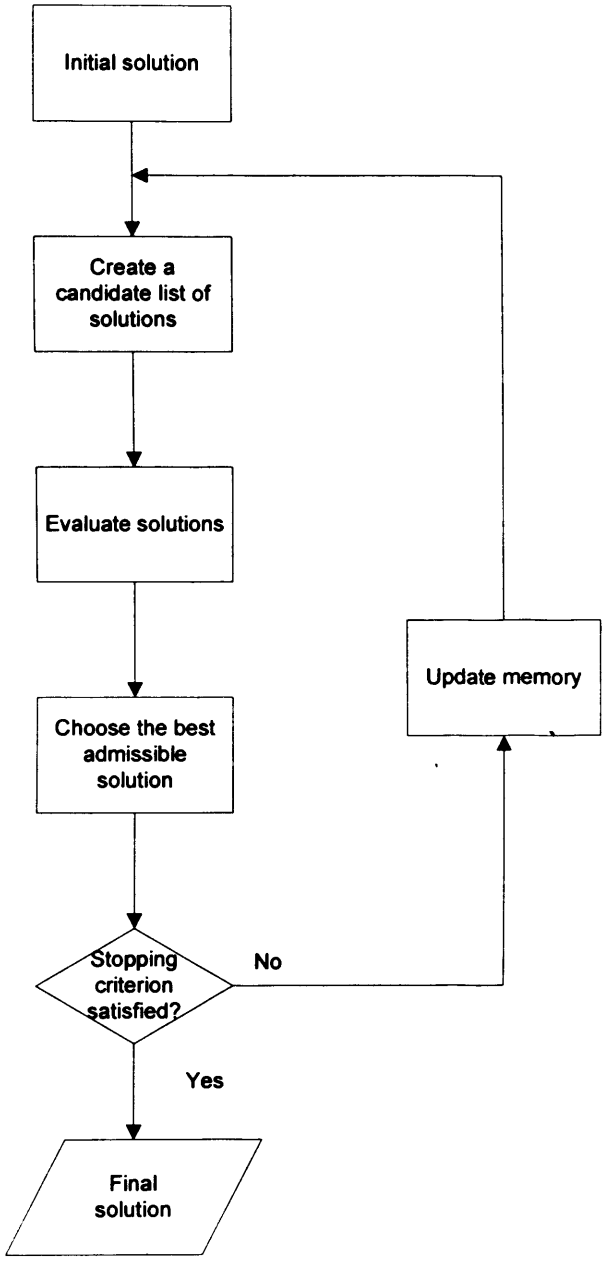


Figure 2.1 A standard Tabu Search

2.4. Simulated Annealing

Simulated Annealing (SA) was introduced by (Kirkpatrick et al. 1983) as a simulation of the annealing of solids to solve combinatorial optimisation problems. The annealing is the process of the slow cooling of preheated solids to achieve a crystalline state. In this algorithm, the optimum solution simulates the state of a perfect crystal and the cost function simulates the energy equation. Figure 2.2 gives a flow chart of the standard SA algorithm, which has been applied to tune fuzzy membership functions (Haber et al. 2009; Liu and Yang 2000), function optimisation (Bohachevsky et al. 1986), to train neural networks (Castillo et al. 1999) and schedule the assembly of PCBs (Hashiba and Chang 1992).

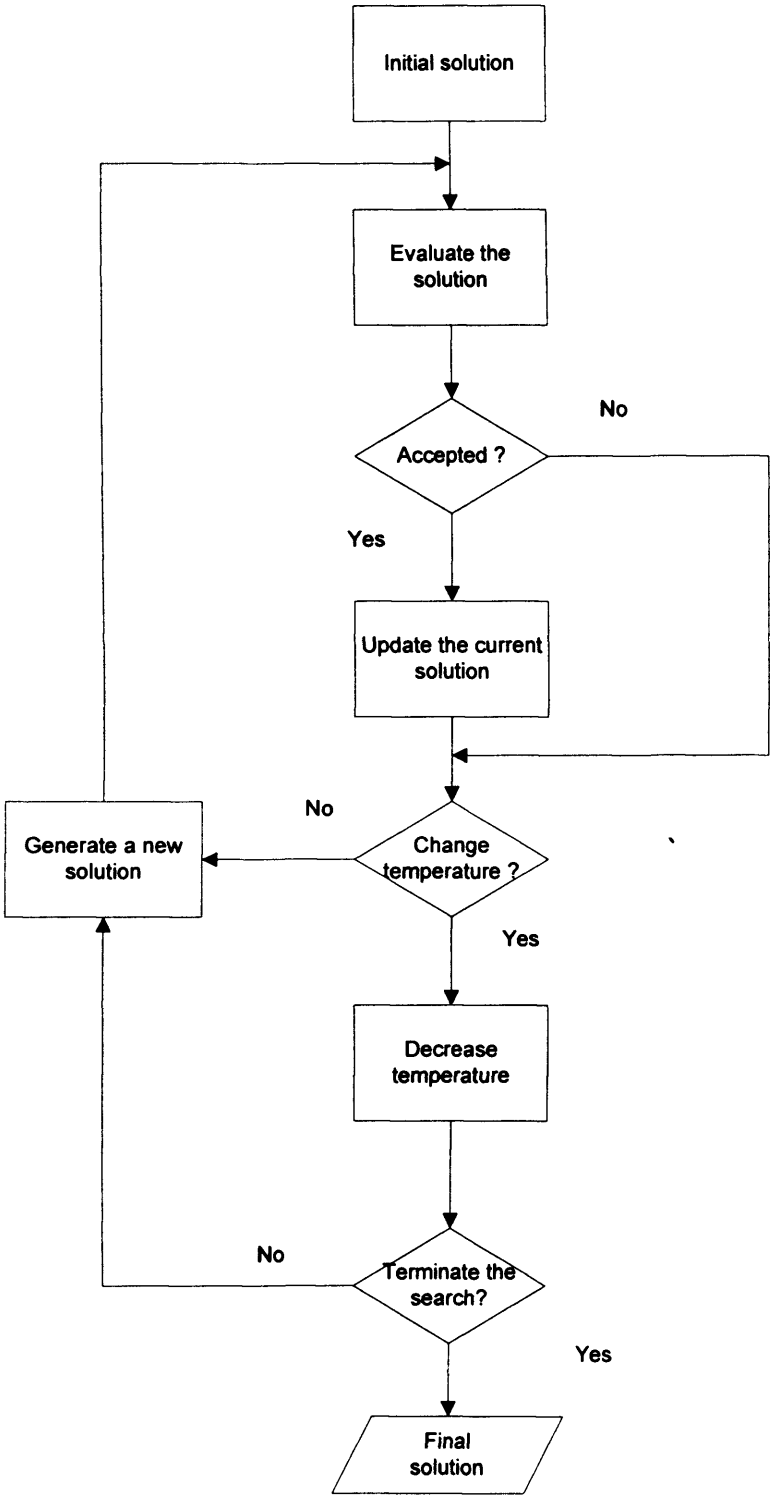


Figure 2.2 A standard Simulated Annealing algorithm

2.5. Artificial Neural Networks

2.5.1. Definition and basic concepts

An Artificial Neural Network (ANN) is a mathematical model of a biological nervous system (Bar-Yam 2003). This model consists of a large number of units or nodes called neurons which are considered simple processing units. Each neuron has its own activation function with a proper threshold (Haykin 1999). Figure 2.3 and Figure 2.4 show a simplified structure of a biological neuron and the structure of an artificial neuron respectively. These nodes or neurons are arranged into layers and connected together by adjustable weights (see Figure 2.5). Generally there are three types of layers: input, output and processing.

The adjustable weights give the capability of learning and adapting the ANN (Pham and Liu 1995). Figure 2.6 illustrates a learning process of an artificial neural network. The different arrangements of the layers and the ways of interconnecting neurons offer a vast number of artificial neural networks with different methods used to tune the adjustable weights (Pham and Liu 1995).

Computational implementations of artificial neural networks are usually presented in high level programming languages (Steeb 2008; Timothy 1993; Valluru and Hayagriva 1995) or in hardware (Eickhoff et al. 2006).

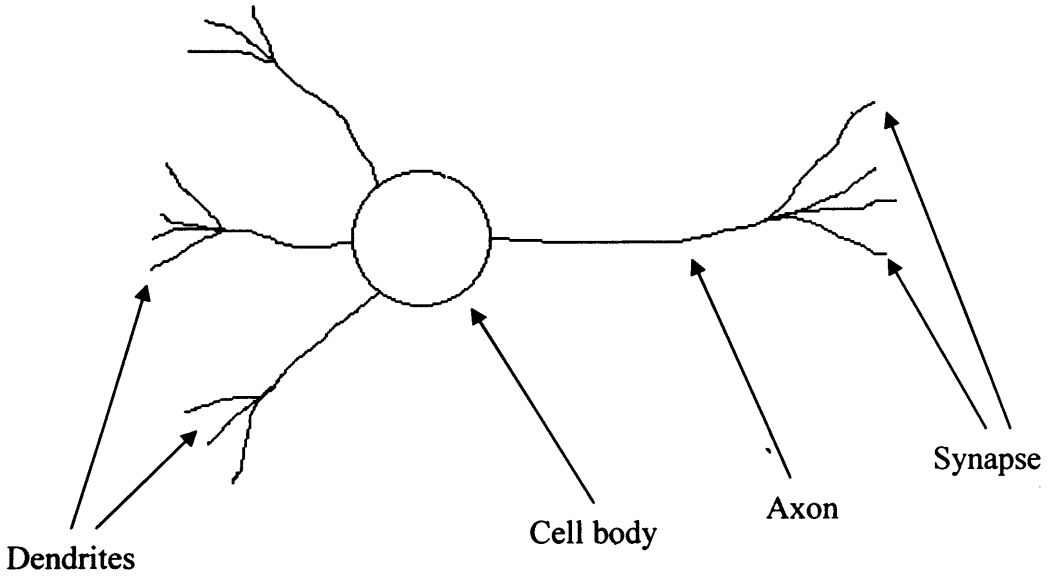


Figure 2.3 A structure of a biological neuron

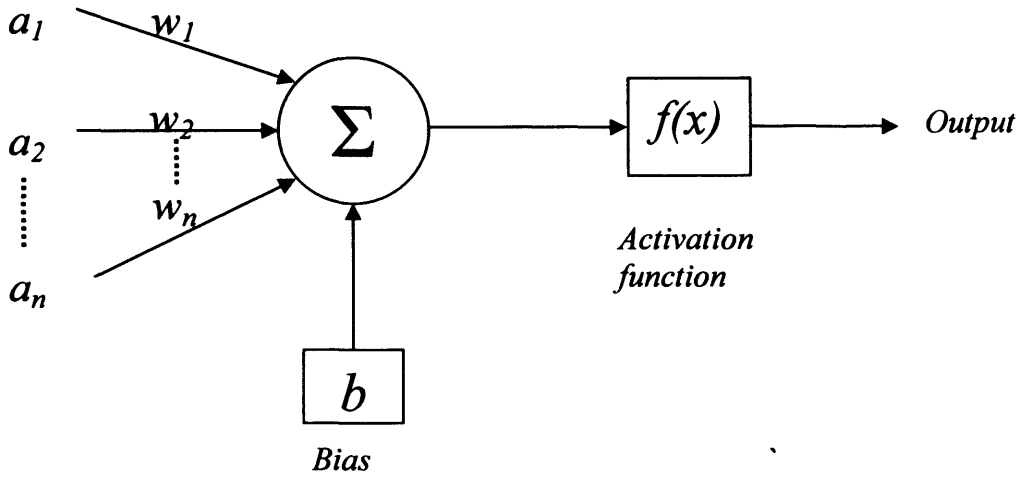


Figure 2.4 The structure of an artificial neuron

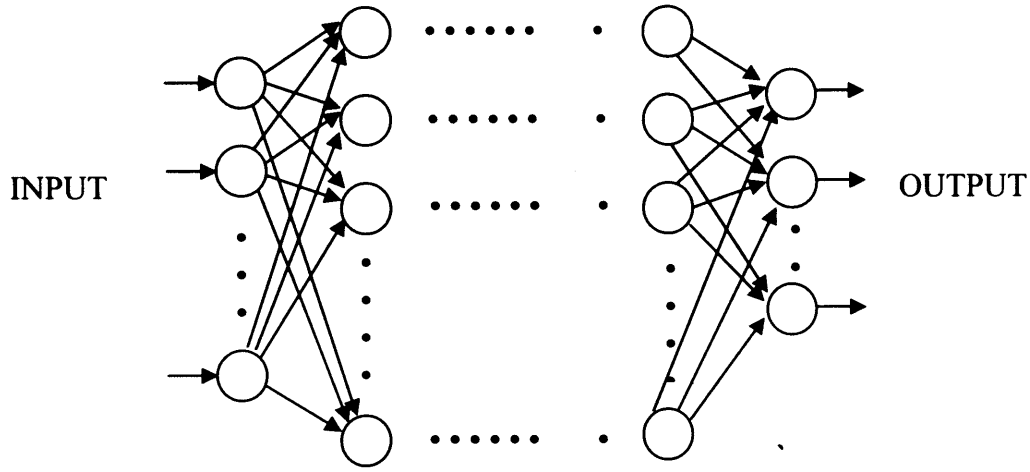


Figure 2.5 A general structure of an artificial neural network

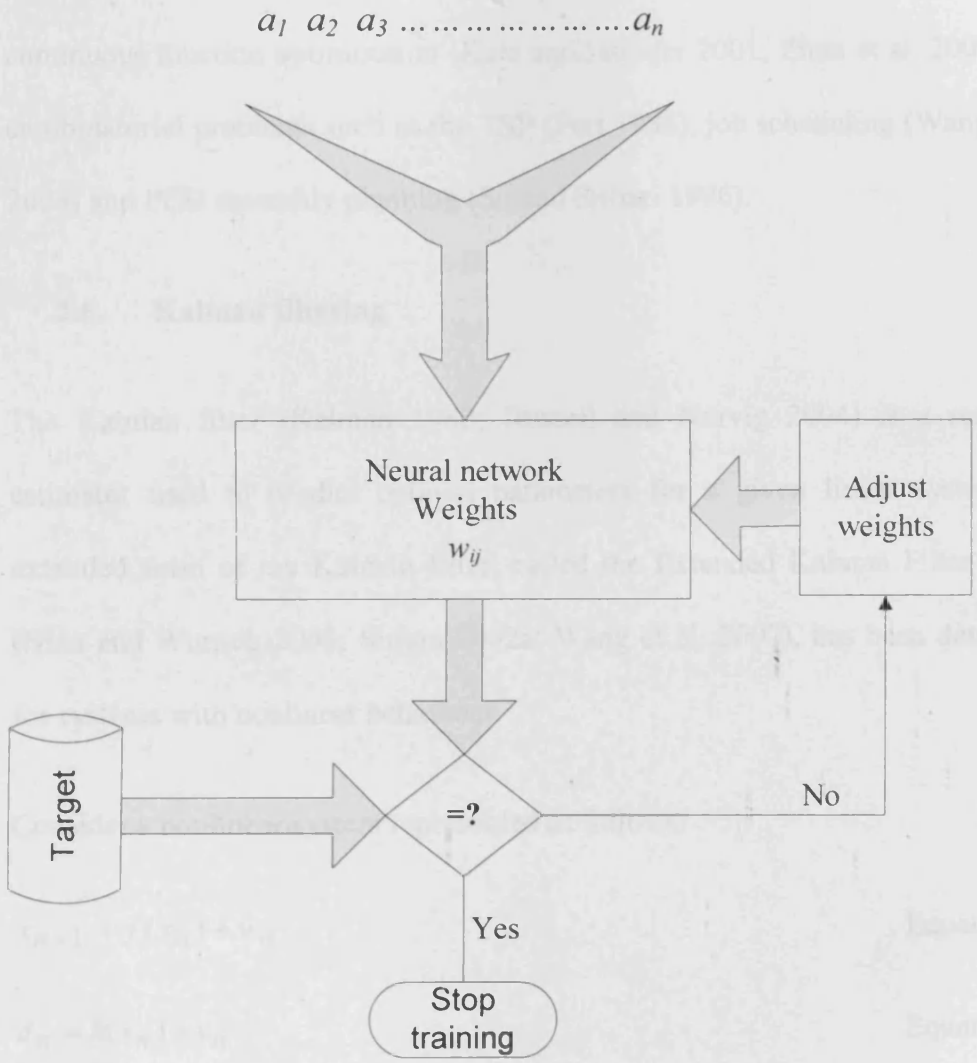


Figure 2.6 A learning process of a neural network

2.5.2. Applications of ANNs in optimisation

ANNs have been applied to a number of optimisation problems such as continuous function optimisation (Kate and Jatinder 2001; Zhou et al. 2007) and combinatorial problems such as the TSP (Fort 1988), job scheduling (Wang et al. 2008) and PCB assembly planning (Su and Srihari 1996).

2.6. Kalman filtering

The Kalman filter (Kalman 1960; Russell and Norvig 2004) is a recursive estimator used to predict optimal parameters for a given linear system. An extended form of the Kalman filter, called the Extended Kalman Filter (EKF) (Nian and Wunsch 2003; Simon 2002a; Wang et al. 2007), has been developed for systems with nonlinear behaviour.

Consider a nonlinear system represented as follows:

$$x_{n+1} = f(x_n) + w_n \qquad \text{Equation 2.1}$$

$$d_n = h(x_n) + v_n \qquad \text{Equation 2.2}$$

where x_n represents the state of the system at time n,

w_n, the process noise,

d_n, the observation vector,

v_n, the observation noise,

$f(\cdot)$ and $h(\cdot)$, nonlinear vector functions of the state.

The following three equations are the recursive estimation equations of the EKF:

$$K_n = P_n H_n (R_n + H_n^{\mathrm{T}} P_n H_n)^{-1} \qquad \text{Equation 2.3}$$

$$\hat{x}_n = f(\hat{x}_{n-1}) + K_n[d_n - h(\hat{x}_{n-1})] \qquad \text{Equation 2.4}$$

$$P_{n+1} = F_n(P_n - K_n H_n^T P_n)F_n^T + Q_n \qquad \text{Equation 2.5}$$

where

$$F_n = \left.\frac{\partial f(x)}{\partial x}\right|_{x=\hat{x}_n} \qquad \text{Equation 2.6}$$

$$H_n^T = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}_n} \qquad \text{Equation 2.7}$$

where K_n is the Kalman gain,

R_n, Q_n are covariance matrices of the noise processes w_n and v_n, respectively,

P_n is the covariance of the prediction error,

$\hat{x}_n$ is the estimated state of the system at time n.

EKF has been used in training neural networks (Ciocoiu 2002; Wang et al. 2007) and tuning fuzzy systems (Nian and Wunsch 2003; Simon 2002a). However, Kalman filtering is very sensitive to the choice of starting point and to parameter tuning, and it is difficult to find proper parameters without extensive trials.

Another problem in employing the Kalman filter as an optimisation tool is that trapping can occur at local optima as the filter tends to converge to local solutions quickly.

2.7. Fuzzy Logic

2.7.1. Fuzzy sets and logic

Boolean logic as a binary logic, with only two states true or false, is easily implemented electronically and computationally due to its simplicity. However, Boolean logic is an inadequate tool to manipulate and process noisy or uncertain measurements. This problem was solved by using overlapped sets without clear boundaries called fuzzy sets (Zadeh 1965). Fuzzy sets and Fuzzy Logic (FL) were introduced by Professor Zadeh in 1965 (Zadeh 1965) as a generalised form of conventional (Boolean) logic and as a mathematical way to represent and manipulate uncertainty in a real world process by means of natural languages.

Fuzzy sets are expressed by Linguistic Variables and Linguistic Values with a value interval [0, 1] instead of the binary state of Boolean logic (Zimmermann 1996). Each Linguistic Value (term) is represented by a membership function. The set of these values together determines how an input variable can be represented within the fuzzy input. The membership functions have many different shapes such as Gaussian, triangular, trapezoidal (Yen and Langari 1999) (see Figure 2.7).

2.7.2. Fuzzy Logic Systems

Fuzzy Logic Systems (FLS) are one of the main developments and successes of fuzzy logic. They are motivated by the biological brain's ability to learn, reason and generalise using noisy or uncertain information (Lei 1999). Mamdani and Assilian (Mamdani and Assilian 1975) introduced the first fuzzy system to control a steam engine with a boiler. Input and output linguistics variables with membership functions and a combination of rules with an inference system were implemented to design a new form of controller with a higher level of abstraction. It was a new development and a complete departure from the traditional approach to the design of controllers.

Sugeno (Sugeno 1985) proposed another type of fuzzy system via simple implementation of mathematical operators. A Sugeno-type system has the same architecture as a Mamdani-type system except for the defuzzification stage. The main difference between them is that the membership functions of output variables in a Sugeno-type system are either linear or constant.

The uncertainty of inputs and outputs of FLSs makes them more noise tolerant than other rule-based systems such as expert systems. Fuzzy systems are able to offer appropriate output in the case of triggering more than one rule at the same time. Another advantage of using fuzzy logic systems is to trim down the

complexity of the required tools and software needed to regulate the outputs of engineering manufacturing works.

A general structure of a fuzzy logic system consists of four units (Dadone 2001; Lee 1990a; Passino and Yurkovich 1998) (see Figure 2.8):

- Fuzzification,
- Defuzzification,
- Inference engine,
- Rule base.

- **Fuzzification**

Fuzzification is the process which translates measured values into real values between 0 and 1. It also assigns these values degrees of truth, usually called membership degree, for the linguistic values of the input linguistics variables.

- **Rule base**

The rule base of a fuzzy logic system consists of a set of fuzzy IF-THEN rules as is illustrated in Figure 2.9. The first terms (after the "IF" and before the "THEN") are called the antecedents of the rules while the last terms (after the "THEN") are the consequences of the rules (Yen and Langari 1999), where x, y and z are linguistic variables and Ai, Bi and Ci are linguistic values of the linguistic variables x, y and z in the universes of discourse U, V and W respectively, with i = 1, 2,…., n.

- **Inference engine (fuzzy reasoning)**

Since fuzzy logic systems are stimulated by the biological brain's capability to make decisions, the inference engine or fuzzy reasoning is considered a method of cloning a human decision making process of judging and giving a proper fuzzy output depending on the inputs and the rule base. Generally, there are two important inference strategies in approximate reasoning (Lee 1990b): generalised modus ponens (GMP) and generalised modus tollens (GMT).

- **Defuzzification**

Defuzzification is the mapping from the linguistic fuzzy output defined over an output universe into a crisp output space (Awadalla 2005). There are many defuzzification strategies; the most common strategies are Maximum, Mean of Maxima and Centroid (Shankir 2000).

In the first strategy, which is the maximum criterion, the maximum membership function value is selected to be the crisp value of the output variable. The second strategy presents the average value of the maximum membership values as the crisp value of the output. Finally, in the Centroid method, which is the most common one, the crisp output is the value of the centre of gravity of the membership functions.

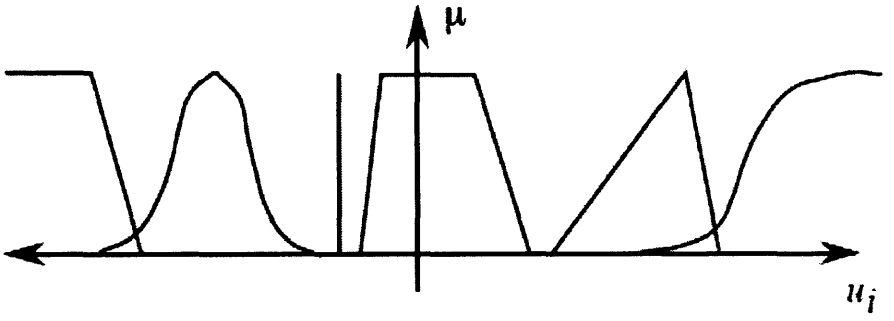


Figure 2.7 Fuzzy membership functions

(Passino and Yurkovich 1998)

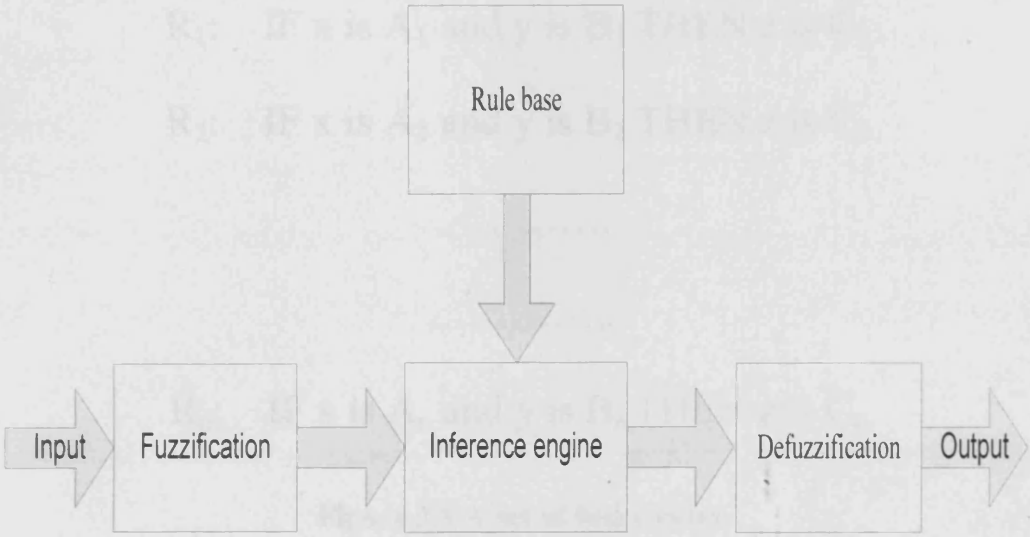


Figure 2.8 A general structure of a fuzzy logic system

R_1: IF x is A_1 and y is B_1 THEN z is C_1

R_2: IF x is A_2 and y is B_2 THEN z is C_2

.....

.....

R_n: IF x is A_n and y is B_n THEN z is C_n

Figure 2.9 A set of fuzzy rules

2.7.3. Applications of fuzzy logic

Fuzzy logic has been implemented in a vast number of applications in many fields such as nonlinear control (Dadam 2002), modelling complex systems (Fahmy 2005), pattern recognition and data clustering (Bezdek 1981) and decision making support systems (Awadalla 2005).

Beside that, fuzzy logic is used in operation research (Narayanaswamy et al. 1996) and for optimisation problems such as job scheduling (Guohua and Yen 1999; Petrovic et al. 2008; Shaout and McAuliffe 1998; Shaout and McAuliffe 2000; Sheibani 2006), planning (Holland 2003; Wang and Chaharbaghi 1995), function optimisation (Pelta et al. 2000), multiobjective optimisation (Reardon 1998; Stanciulescu et al. 2003) and energy saving (Martinsen and Krey 2008; Silva et al. 2002).

2.8. Genetic Algorithm

The Genetic Algorithm (GA) was proposed in the 1960s by Holland (Holland 1975, 1992) and was inspired by the natural selection and evolution theory proposed by Charles Darwin and the theoretical background of "schema theorem". This inspiration has been implemented in operators used to improve the fitness of the individuals of the population generation by generation.

The GA is an iterative population-based algorithm where each iteration represents a generation. The GA usually manipulates individuals as binary-coded strings. This string is likened to a chromosome, with substrings called genes. Each parameter of the problem (each dimension of the search space) is represented by a binary substring (gene) (Pham and Karaboga 1999).

The initialisation status of the population is generated by assigning randomly independent samples from the search space to each individual of the population. The individuals are then evaluated and given a fitness value via an objective function. Afterwards, selection is made for reproduction (to form a mating pool). In the basic form of the GA, selection is proportional to fitness (roulette wheel) to ensure that better individuals have higher chance to be selected (Goldberg 1989).

Genetic operators (Crossover and Mutation) are applied to generate new samples from a search space. In Crossover, new individuals are generated by mating existing selected individuals. Crossover is performed by swapping parts of two existing parents to produce two new children.

Another operator is Mutation, where new individuals are generated by random bits inversion with a specified rate for the code of all individuals. Figure 2.10 shows a flowchart of a simple GA.

2.8.1. Applications of Genetic Algorithm

GA has been used to solve a large number of optimisation problems in the continuous domain such as tuning fuzzy logic controllers (Herrera et al. 1995; Lee and Smith 1994), neural network training (Rooij et al. 1996) and combinatorial optimisation problems such as the Travelling Salesman Problem (TSP) (Braun 1990; Takahashi 2005), job scheduling (Kamrul Hasan et al. 2007; Lawrence 1985), vehicle routing (Baker and Ayechew 2003) and PCB assembly planning (Garcia-Naijera and Brizuela 2005; Ho and Ji 2005, 2006; Khoo and Loh 2000; Khoo and Ng 1998; Leu et al. 1993; Maimon and Brha 1998; Ong and Khoo 1999; Wong and Leu 1993).

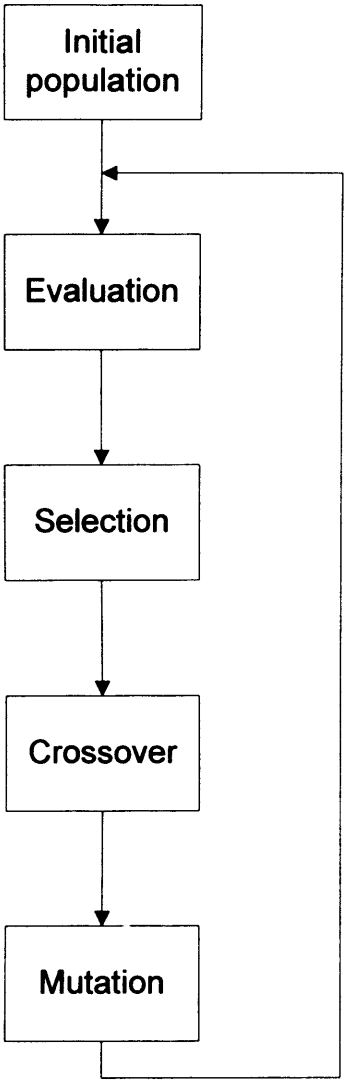


Figure 2.10 A flowchart of a simple Genetic Algorithm

2.9. Ant Colony Optimisation

Ant Colony Optimisation (ACO) is a swarm-based algorithm inspired by real ants and attempts to mimic their natural foraging behaviour. It was proposed as a novel algorithm to solve combinatorial optimisation problems (Dorigo et al. 1996).

Real ants use an indirect communication method among themselves while they forage for food. This search begins with a random exploration of the environment around the colony nest. When any ant comes across a food source, it takes some of the food to the colony nest and puts down a trace of a natural chemical material called pheromone. The placed pheromone helps other ants to locate the food source (Dorigo and Blum 2005). One of the characteristics of the pheromone is its evaporation with time. This effect reduces the quantity of pheromone deposited on the path to the food source, so the greater the pheromone, the shorter time since the food source was located. Thus the quantity of remaining pheromone gives an idea about the quality of the path to the food source (length of the path). This behaviour was modelled computationally in the ACO algorithm.

The task of the ACO is to find an optimum sequence of parameters in a combinatorial problem to reduce the cost function, where the sequence of

parameters is likened to a path with several nodes, each node corresponding to one of the solution's parameters.

Moving from one node to another is given probabilistically by Equation 2.8.

$$P_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum \tau_{iu}^{\alpha} \eta_{iu}^{\beta}} \qquad \text{Equation 2.8}$$

where τ_{ij} represents the *a posteriori* effectiveness of the move from node i to node j,

η_{ij} represents the *a priori* effectiveness of the move from i to j,

α is a parameter to control the influence of τ_{ij},

β controls the influence of η_{ij}.

Pheromone concentration on each link (i, j) is updated by using Equation 2.9.

$$\tau_{ij} = \rho \tau_{ij} + \Delta \tau_{ij} \qquad \text{Equation 2.9}$$

where ρ is the rate of pheromone evaporation and $\Delta\tau ij$ is the amount of pheromone deposited.

Figure 2.11 shows the pseudo code of ACO algorithm.

1- Procedure ACO_MetaHeuristic.

2- While (stopping criterion not met)

3- Generate solutions

4- Pheromone update using Equation 2.9

5- Daemon Action, move according to probability calculated with Equation 2.8

6- End While

7- End Procedure

Figure 2.11 Pseudo code of simple ACO

2.9.1. Applications of Ant Colony Optimisation

ACO has been applied to a variety of combinatorial problems such as the TSP (Bontoux and Feillet 2008; Cheng and Mao 2007; Dorigo and Gambardella 1997; Shang et al. 2007), job scheduling (Jain and Sharma 2005; Seo and Kim 2009) and vehicle routing problems (Bell and McMullen 2004; Mazzeo and Loiseau 2004). Recently, a number of developments have been applied to ACO to make it be more suitable for continuous optimisation problems (Mathur et al. 2000; Yu et al. 2007).

2.10. Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) was introduced in 1995 by Eberhart and Kennedy (Eberhart and Kennedy 1995; Kennedy and Eberhart 1995) to mimic the flocking behaviour of a swarm of living creatures such as birds, insects or fish. Similar to the ACO and GA, PSO is a swarm-based algorithm which consists of a group of individuals acting collectively to find an optimum. The individuals communicate either directly or indirectly with one another in each search direction.

In PSO, the number of the individuals stays fixed during the search process. Each individual is called a particle and is supplied with a velocity and a position. Every one of the particles has a memory function to store the best position that it has so

far visited (local best) and the overall best position achieved by the whole swarm (global best).

New parameters have been added to enhance the performance of the basic form of PSO algorithm, such as inertia weight (Engelbrecht 2005; Shi and Eberhart 1998a, b). The inertia weight is implemented in the update equation of velocity of each particle according to Equation 2.10.

$$\overrightarrow{v_i} = w\overrightarrow{v_i} + c_1 \overrightarrow{\varphi_{1i}}\left(\overrightarrow{p_i} - \overrightarrow{x_i}\right) + c_2 \overrightarrow{\varphi_{2i}}\left(\overrightarrow{p_g} - \overrightarrow{x_i}\right) \qquad \text{Equation 2.10}$$

where w is the *inertia weight*,

$\overrightarrow{p_i}$ and $\overrightarrow{p_g}$ are the *local best* and *global best* respectively,

φ_1 and φ_2 are random numbers between (0,1),

c_1 and c_2 are acceleration coefficients to control the maximum step size the particle can achieve.

The position of each particle is updated at each iteration by adding the velocity vector to the position vector according to Equation 2.11.

$$\overrightarrow{x_i} = \overrightarrow{x_i} + \overrightarrow{v_i} \qquad \text{Equation 2.11}$$

The inertia weight w and the acceleration coefficients c_1 and c_2 regulate the velocity update of each particle. A proper selection of the inertia weights and the acceleration coefficients can provide an equilibrium between the global and the local search, since a large inertia weight value leads to global exploration in spite of local exploitation with a small inertia weight (Engelbrecht 2005).

2.10.1. Applications of Particle Swarm Optimisation

PSO has been used in both forms of optimisation problems (contentious and combinatorial). PSO has been applied to neural network training (Gudise and Venayagamoorthy 2003; Kennedy 1997; Pham and Sholedolu 2006), fuzzy system learning (Feng 2005), neuro-fuzzy system (Zhao and Yi 2006), TSP (Shi et al. 2007; Wei et al. 2004) and job scheduling (Abraham et al. 2006; Tu et al. 2006).

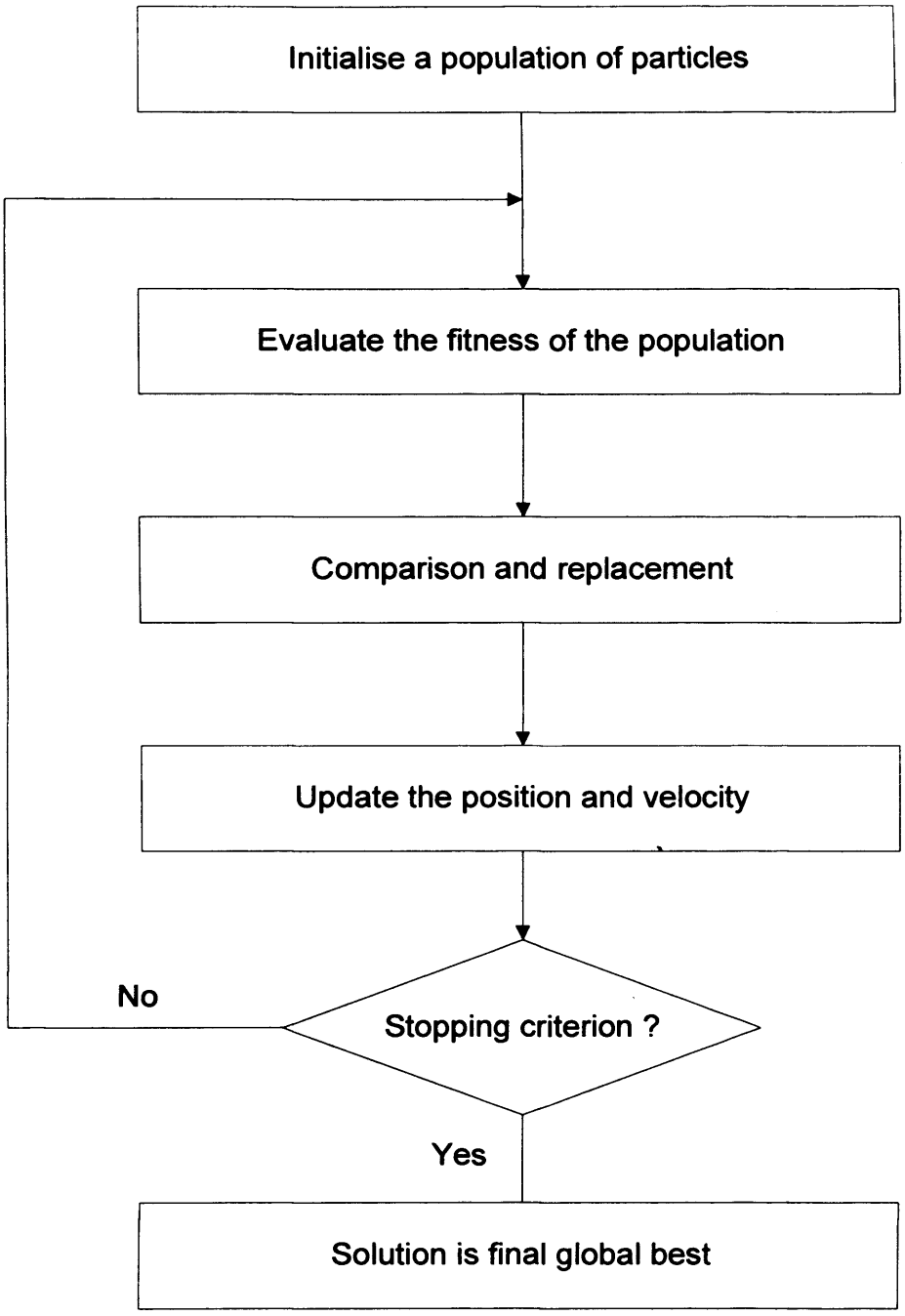


Figure 2.12 A flow chart of PSO

2.11. Bees-inspired algorithms

Bees-inspired algorithms are motivated by the natural behaviour of swarms of bees (Yang 2008). Foraging behaviour (Seeley 1996) and nesting site selection (Passino et al. 2008) have been modelled computationally to be used as optimisation methods in either combinatorial or continuous search space.

The honey bee algorithm was proposed in 2004 (Tovey 2004) and applied to internet server optimisation. The BeeHive algorithm (Wedde et al. 2004) was introduced and applied to routing problems in packet switching networks (Muddassar 2008) where agents called BeeAgents are used to route packets among network nodes.

Another implementation of bee behaviour was presented by (Teodorovic and Dell'orco 2005) to solve transportation problems and was called Bee Colony Optimisation. This algorithm is a constructive approach which is similar to ACO. Later, Virtual Bees Algorithm (VBA) (Yang 2005) was announced as a model of the natural foraging behaviour of honey bees. It is supplied with PSO-like parameters and applied to continuous optimisation. Artificial Bees Colony (ABC) algorithm (Karaboga and Basturk 2008) is another optimisation tool inspired by the foraging behaviour of honey bees that has been applied to continuous optimisation problems. While Quijano and his colleague (Quijano and Passino

2007a, b) have proposed a model of honey bee social foraging to form the basis of an algorithm to solve optimal resource allocation problems.

2.12. The Bees Algorithm

The Bees Algorithm is inspired by the natural foraging behaviour of honey bees to solve complex optimisation problems. It was proposed by Professor Pham and his colleagues (in MEC, Cardiff University) in 2005 (Pham et al. 2005; Pham et al. 2006b). The Bees Algorithm is considered the first general purpose bees-inspired algorithm.

2.12.1. Foraging Behaviour of honey bees

Honey bees naturally use a special form of *foraging behaviour* to identify sources of food and to collect it, in a situation where groups of honey bees harvest many food sources simultaneously. Within the foraging process, the population of the honey bees' hive is divided into two types (i) scout bees and (ii) worker bees.

The collection of food, or harvesting, begins by sending scout bees around the area of the hive to explore randomly the environment surrounding the hive to find food sources (Seeley 1996). When the scouts return to the hive they carry information about the area surrounding the hive, and show the locations of patches of flowers and the quantity of nectars in them by performing a dance called the *Waggle Dance* (Seeley 1996) on a "dance floor" at the entrance to the

hive. This dance is essential for honey bees to pass information from scout bees to worker bees regarding a flower patch: the direction in which it will be found, its distance from the hive, and its quality rating (or fitness) (Camazine et al. 2003).

After the *Waggle Dance*, a number of worker bees fly to flower patches to collect nectar. More worker bees go to the more promising flower patches (Camazine et al. 2003) and at the same time scout bees continue to explore for more promising flower patches.

2.12.2. The algorithm

A basic form of the Bees Algorithm was proposed to follow the natural foraging behaviour of honey bees to find optima. The algorithm uses uniformly distributed random search for global exploration and local exploitation. The algorithm manipulates data as floating point numbers instead of the binary-coding of the search space as used in genetic algorithms (Pham and Karaboga 1999).

A number of parameters need to be preset to run the algorithm, including; the number of scout bees (n), number of patches selected for the local search (m), number of elite patches among m selected patches (e), number of worker bees to be recruited for the elite e patches (nep), number of worker bees to be recruited for the other (m-e) selected patches (nsp), the initial size of each patch (ngh) and the stopping condition.

Figure 2.13 and Figure 2.14 shows the pseudo code and flowchart of the basic Bees Algorithm. Figure 2.15 gives a graphical explanation of the basic Bees Algorithm.

An improved form of the Bees Algorithm was introduced by (Ghanbarzadeh 2007) in his thesis with interpolation and extrapolation mating of the unselected bees. The shrinking method for neighbourhood size, "abandon" when stuck in a local optimum, and "abandon sites without new information" were also proposed.

A hybrid PSO-Bees Algorithm (Pham and Sholedolu 2008) was proposed to solve the problem of premature convergence in the basic PSO algorithm. In the hybrid PSO-Bees Algorithm, adaptive neighbourhood search (shrinking method) and random particles were added to global search.

1- Initialise population with random solutions.

2- Evaluate fitness of the population.

3- While (stopping criterion not met).

//Forming new population.

4- Select patches for neighbourhood search.

5- Recruit bees for selected patches (more bees for best e patches) and evaluate their fitness.

6- Select the fittest bee from each patch.

7- Assign remaining bees to search randomly and evaluate their fitness.

8- End While.

Figure 2.13 Pseudo code of the basic Bees Algorithm

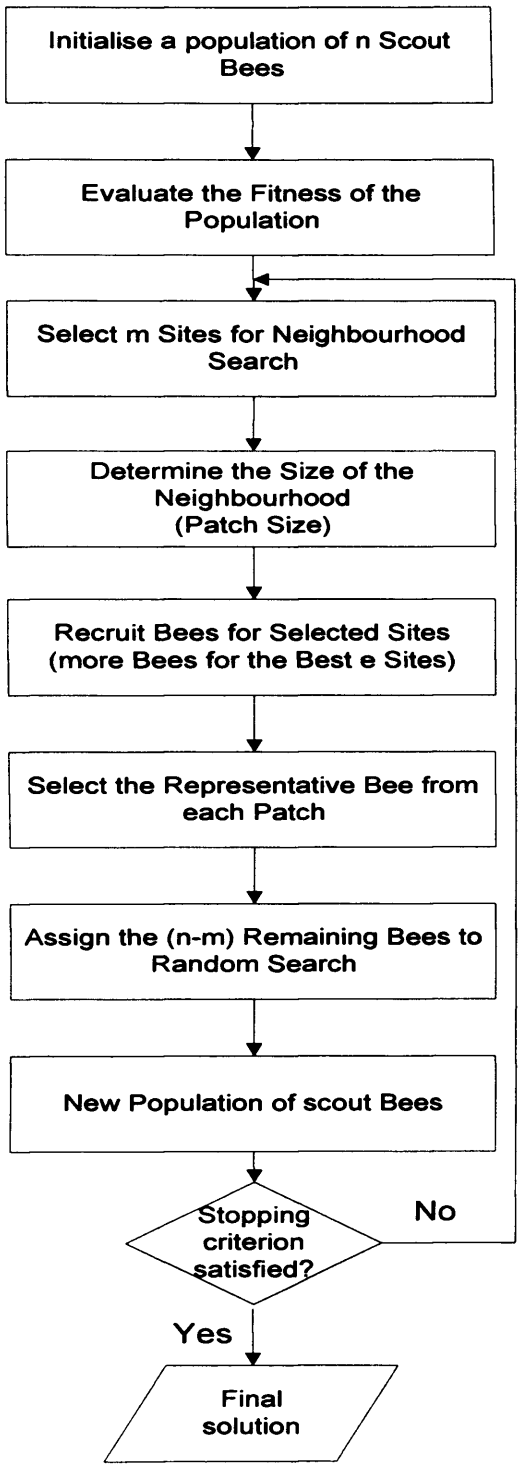


Figure 2.14 The flowchart of the basic Bees Algorithm

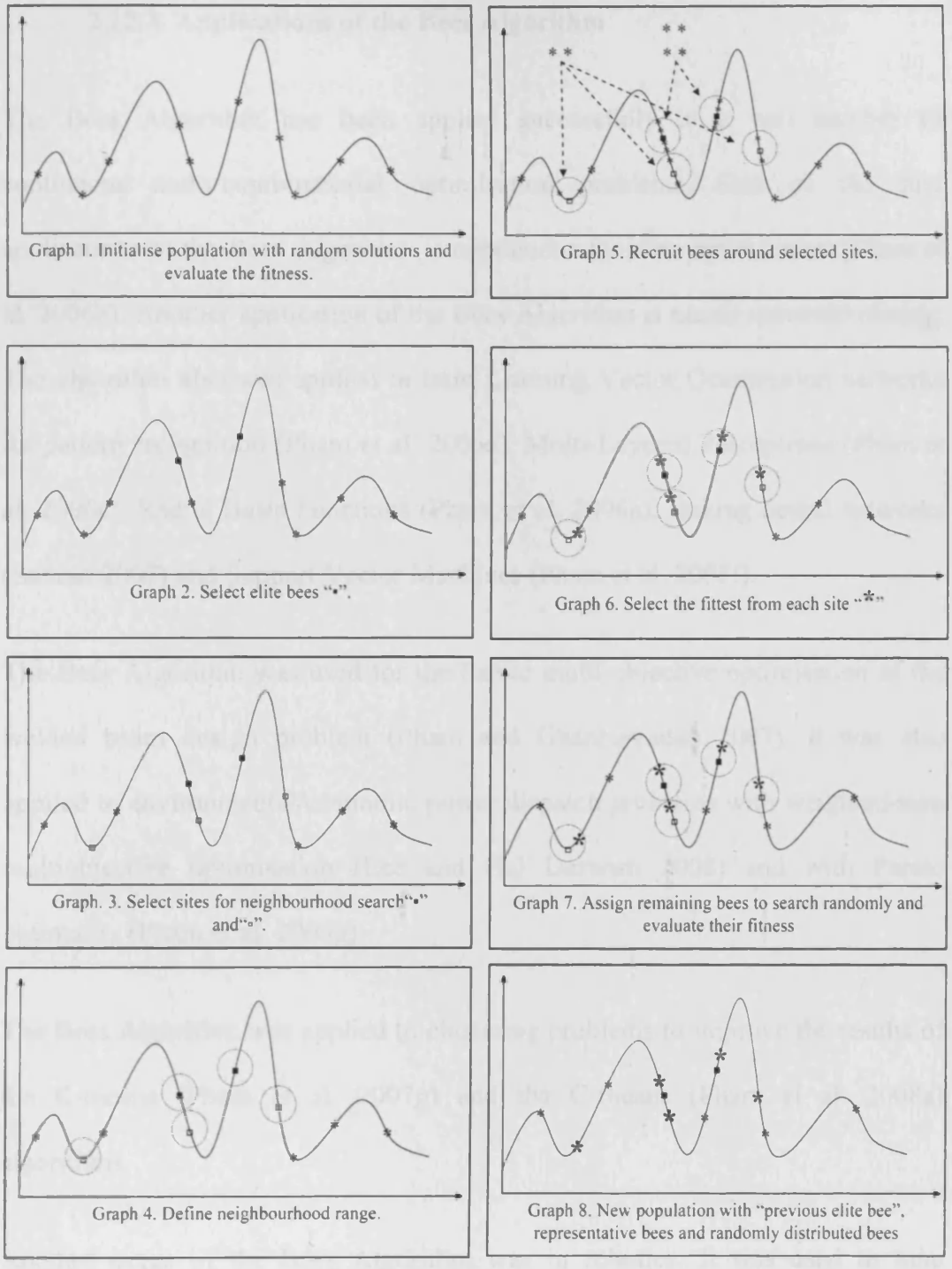


Figure 2.15 Graphical illustration of the Bees Algorithm

(Ghanbarzadeh 2007)

2.12.3. Applications of the Bees Algorithm

The Bees Algorithm has been applied successfully to a vast number of continuous and combinatorial optimisation problems. One of the first applications of the Bees Algorithm is continuous function optimisation (Pham et al. 2006b). Another application of the Bees Algorithm is neural network training. The algorithm also was applied to train Learning Vector Quantisation networks for pattern recognition (Pham et al. 2006d), Multi-Layered Perceptrons (Pham et al. 2006c), Radial Basis Functions (Pham et al. 2006a), spiking neural networks (Sahran 2007) and Support Vector Machines (Pham et al. 2007f).

The Bees Algorithm was used for the Pareto multi-objective optimisation of the welded beam design problem (Pham and Ghanbarzadeh 2007). It was also applied to environmental/economic power dispatch problems with weighted-sum multiobjective optimisation (Lee and Haj Darwish 2008) and with Pareto optimality (Pham et al. 2008g).

The Bees Algorithm was applied to clustering problems to improve the results of the K-means (Pham et al. 2007g) and the C-means (Pham et al. 2008a) algorithms.

Another usage of the Bees Algorithm was in robotics. It was used to tune Proportional-Integral-Derivative (PID) control parameters of a flexible robot

manipulator (Pham et al. 2008f), PID control (Jones and Bouffet 2008) and for learning the inverse kinematics of a robot manipulator (Pham et al. 2008c).

Preliminary design is another application of the Bees Algorithm (Pham et al. 2007c). The algorithm has been used to generate branded product concepts (Pham et al. 2008b) and to design mechanical components (Pham et al. 2008d; Pham et al. 2009; Pham et al. 2007i).

The algorithm was also used to obtain the optimal sink path for large-scale sensor networks (Saad et al. 2008a; Saad et al. 2008b). It was also applied to design a reconfigurable dual-beam linear antenna array (Guney and Onay 2008) and the pattern synthesis of linear antenna arrays (Guney and Onay 2007).

The Bees Algorithm has been implemented in discrete space for combinatorial problems such as manufacturing cell formation (Pham et al. 2007a), job scheduling (Pham et al. 2007d), PCB assembly planning (Pham et al. 2007h) and time tabling (Lara et al. 2008). Feature selection (Pham et al. 2007e) was another task for which the algorithm was used.

The Bees Algorithm also has been used in a chemical engineering process as a dynamic optimisation tool (Pham et al. 2008h) and in biology computing (Bahamish et al. 2008).

2.13. Conclusion

The Bees Algorithm as an intelligent optimisation technique was inspired by the natural foraging behaviour of honey bees to solve complex optimisation problems. It was introduced to solve both continuous and combinatorial problems. However, it suffers from a number of disadvantages such as the large number of tuneable parameters needed to run the algorithm and the slow convergence of the local search part.

In this work, the advantage of fuzzy logic to trim down the complexity of modelled systems will be used to simplify the usage of the Bees Algorithm. Fuzzy logic will be implemented to reduce the number of parameters needed to run the algorithm.

The fast convergence of the Kalman filter to local optima will also be exploited to construct an efficient method to update the positions of worker bees in the local search part of the Bees Algorithm.

CHAPTER 3. USING FUZZY LOGIC TO ENHANCE THE BEES ALGORITHM

3.1. Preliminaries

The Bees Algorithm was developed to mimic the food foraging mechanisms which are found in honey bee swarms and to use this behaviour as a model for an optimisation algorithm. Such natural behaviour is the result of millions of years of natural evolution.

Recent literature (Pham et al. 2005; Pham et al. 2006b) shows that the basic form of the Bees Algorithm needs a large number of tuneable parameters to be set to run the algorithm.

This chapter shows an enhanced form of the Bees Algorithm which implements a fuzzy logic system for the greedy selection of local search sites. The proposed fuzzy greedy selection system reduces the numbers of parameters needed to run the algorithm.

3.2. The Selection process in the Bees Algorithm

3.2.1. Greedy selection in the basic Bees Algorithm

Greedy algorithms (Russell and Norvig 2004) are natural and usually simple and fast. As its name implies, a greedy optimisation algorithm builds a solution by using the best possible choices. The natural foraging behaviour of honey bees (Seeley et al. 2006) also is greedy, where honey bees tend to recruit more worker bees for the best possible patches.

This greedy selection of flower patches is a feature of greedy algorithms which have been used to solve optimisation problems and always make the choice that looks best at that moment. A greedy algorithm leads to an optimal solution just as a locally optimal choice leads to a globally optimal solution (Russell and Norvig 2004).

The basic Bees Algorithm uses greedy selection to choose (m) best and (e) elite patches respectively out of the explored patches by scout bees. The visited patches are usually set in a descending sorted list according to their fitness value to form a candidate list.

In the basic Bees Algorithm again, the selection from the candidate list is related to the rank of each visited patch. The parameters of the selection process are chosen empirically by the user to determine (e) elite patches and (m) best patches beside the rest of the parameters (nep), (nsp) and (ngh). The mentioned

parameters (e), (m), (n), (nep), (nsp) and (ngh) need extra tuning efforts from a user to set the best possible combination.

Moreover, hard threshold selection may ignore some promising search sites because their ranks are less than the selection threshold (m) or (e). An alternative selection process is proposed in this chapter called fuzzy greedy selection, which implements a fuzzy logic system as a decision making support system to choose local search sites and recruit worker bees inside them.

3.2.2. The proposed fuzzy greedy selection system

Fuzzy logic has been introduced to represent vague information such as linguistics variables with fuzzy membership functions without hard thresholds and unlike classical logic, which requires a deep understanding of a system, exact equations and precise numeric values.

Fuzzy logic allows the expression of the knowledge with subjective concepts such as "high" and "low" which are mapped into exact numeric ranges. Beside the subjective concepts, the uncertainty of inputs and outputs of fuzzy logic systems allows better modelling to mimic the natural foraging behaviour of honey bees.

A fuzzy greedy selection system was constructed to decide the number of the selected patches and the number of recruited workers for each selected patch. It is

a form of fuzzy decision making system which tends to eliminate patches with low fitness or low rank. From now on there will be no need to preset the values of (m), (e), (nep) and (nsp), since they are determined automatically by the fuzzy greedy selection system.

The system consists of two fuzzy inputs, two constants for output, and a Sugeno inference system with 4 fuzzy rules (see Figure 3.1). A typical fuzzy rule in a Sugeno model (Jang et al. 1997) has the form:

if x is A and y is B then z = f(x, y) Equation 3.1

where $z=f(x,y)$: a crisp function in the consequent.

$f(x,y)$: a polynomial function; but it can be any function.

In the 1st-order Sugeno fuzzy model: $f(x, y)$ is a 1st order polynomial, while in the zero-order Sugeno fuzzy model: $f(x,y)$ is a constant. Figure 3.2 shows a typical Sugeno fuzzy model.

The inference system used is a zero-order Sugeno fuzzy model (a special case of Mamdani model) with weighted average for aggregation. A Sugeno model is used in the proposed algorithm in view of the fact that it provides a more compact and computationally efficient representation than a Mamdani model.

Each one of the input variables consists of two triangular membership functions called "low" and "high". The first input variable is the fitness value and the second input variable is the rank value. Figure 3.3 illustrates the shape of the membership functions used for the input variables.

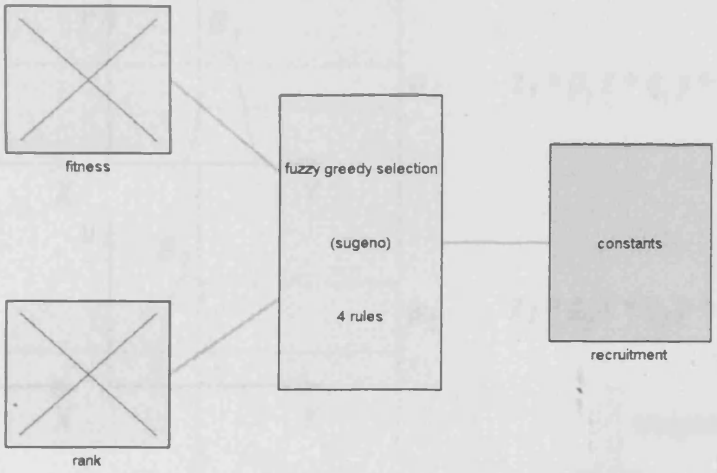


Figure 3.1 Fuzzy greedy selection: 2 inputs, 1 output and 4 rules

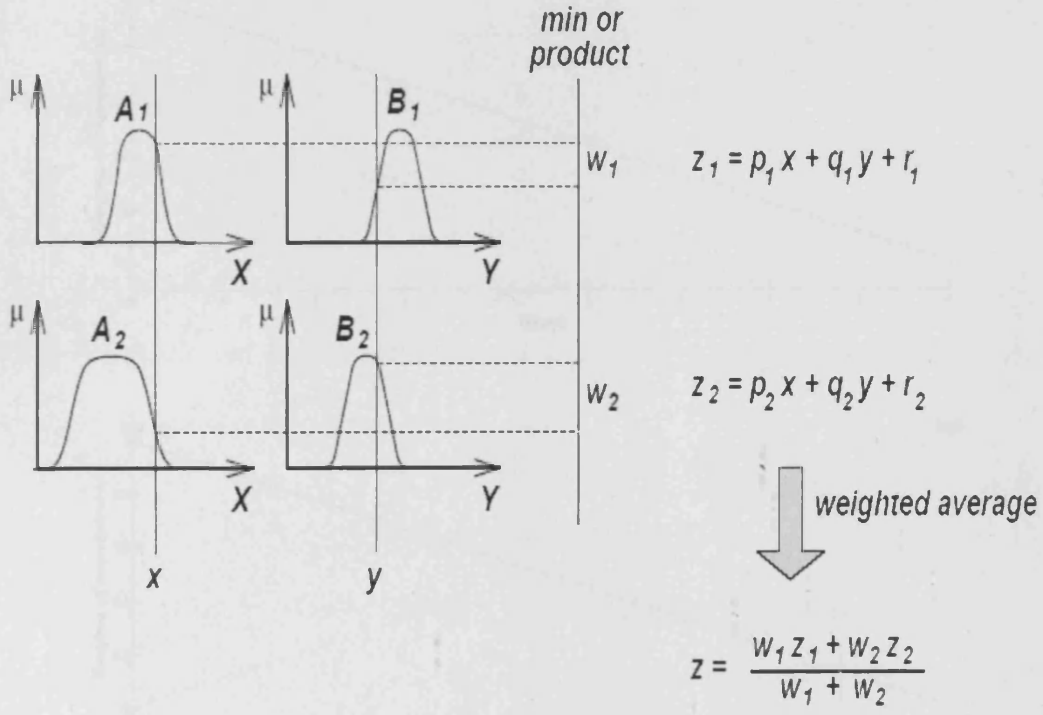


Figure 3.2 Sugeno fuzzy model

(Jang et al. 1997)

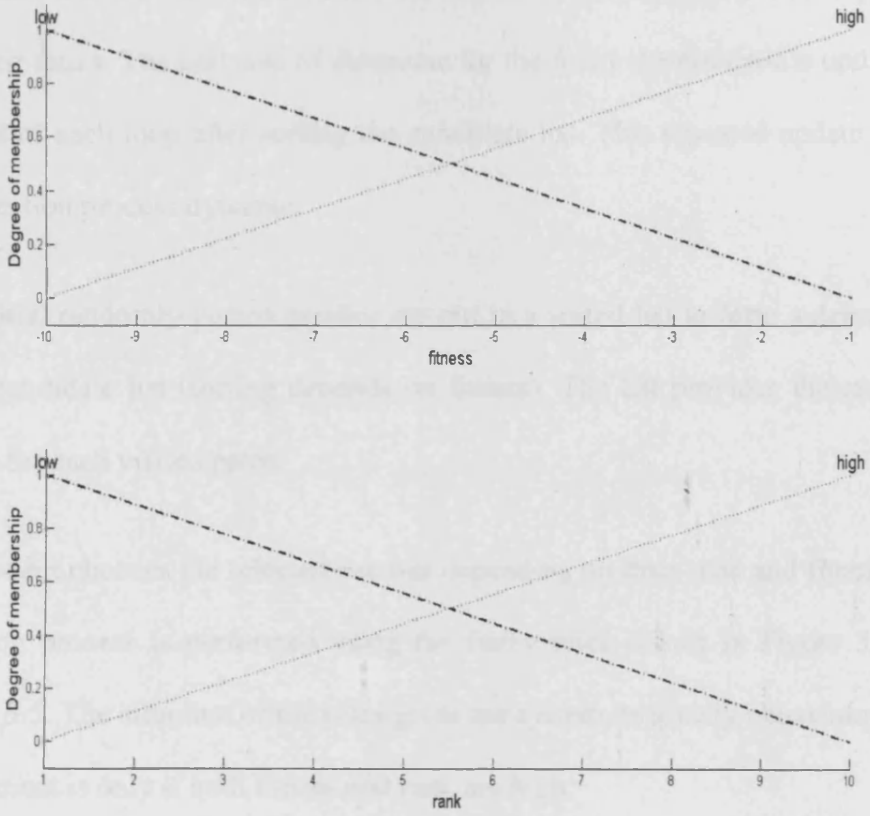


Figure 3.3 The membership functions of the input variables

The output consists of two constants: low with value (0) zero and high with value (nw), where (nw) is the maximum number of worker bees in each patch.

The initial universe of discourse for the inference system used comes from the maximum and the minimum values for fitness of the randomly visited patches and their ranks. The universe of discourse for the fuzzy system used is updated at the end of each loop after sorting the candidate list. This repeated update makes the selection process dynamic.

The initial randomly visited patches are put in a sorted list to form a descending order candidate list (sorting depends on fitness). The list provides the rank and fitness for each visited patch.

The system chooses the selected patches depending on their rank and fitness. The selection process is performed using the fuzzy rules shown in Figure 3.4 and Figure 3.5. The structure of the rules gives the system its greedy behaviour, since recruitment is only if both fitness and rank are high.

The output of the fuzzy system is rounded to give the total number of worker bees in a selected patch. Figure 3.6 illustrates the representative surface of the fuzzy system.

1. If (fitness is low) and (rank is low) then (recruitment is low)

2. If (fitness is low) and (rank is high) then (recruitment is low)

3. If (fitness is high) and (rank is low) then (recruitment is low)

4. If (fitness is high) and (rank is high) then (recruitment is high)

Figure 3.4 Fuzzy rules of the greedy selection system

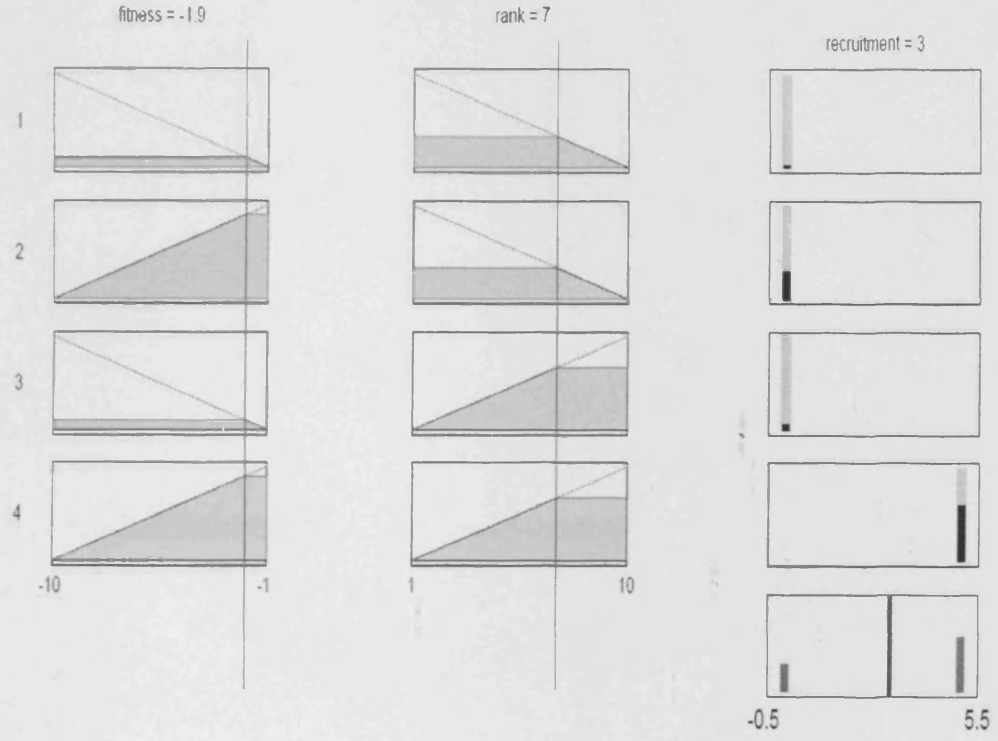


Figure 3.5 Fuzzy rules of the selection system

In the third column (recruitment), dark colour represents the number of recruited bees for a selected site. While light colour is the highest number of worker bees (nw) which is chosen by the user.

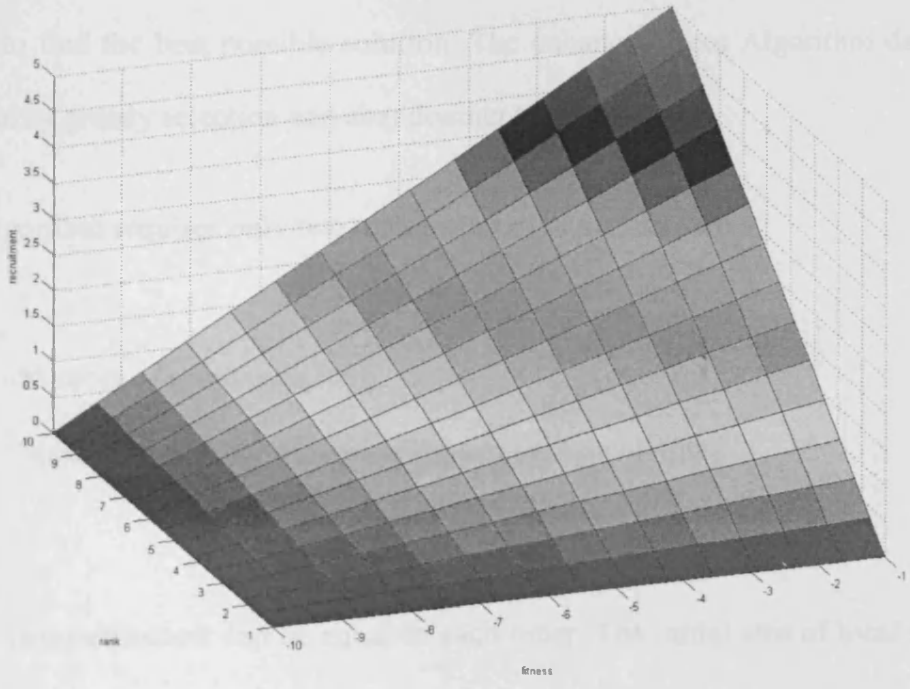


Figure 3.6 The representative surface of the fuzzy system

3.3. The enhanced Bees Algorithm with fuzzy selection

The proposed algorithm is called the enhanced Bees Algorithm. This algorithm implements the foraging strategy of honey bees which involves finding a flower patch, deciding whether to enter it and search for food and when to leave it.

The proposed algorithm uses the basic Bees Algorithm as a core, where a bee seeks to find the best possible solution. The enhanced Bees Algorithm depends on a fuzzy greedy selection and abandonment process.

The algorithm requires only two parameters to be set, namely:

- Number of scout bees (ns);
- Maximum number of worker bees in each patch (nw).

These two parameters can be equal to each other. The initial size of local search patches (ngh) is a percentage of the size of the search space.

In the enhanced Bees Algorithm each explored patch is assigned with a local memory. This memory is shared among worker bees in a patch. It stores the best fitness of a patch and the recent patch size.

Figure 3.7 and Figure 3.8 show the pseudo code and the flowchart of the enhanced Bees Algorithm, respectively.

Similar to all population-based algorithms, the enhanced Bees Algorithm starts with (ns) scout bees being placed randomly in the search space. The fitnesses of the sites visited by the scout bees are evaluated in step 2.

In step 3, a fuzzy greedy system is formed and initiated with the values of fitness and rank of the visited sites (patches) by scouts. In the proposed algorithm, the radii of the neighbourhood search (ngh) (local search site) are proportional to the search space dimension intervals (see Figure 3.9).

In steps 5 and 6, the fuzzy greedy system conducts the selection and recruitment of worker bees around selected sites. Selection and recruitment are made according to the fitness and rank associated with each site (more bees for higher fitness and rank). This method for selection and recruitment is performed in a differentially smooth manner without any hard threshold.

In step 7, for each site, only the fittest bee with highest fitness will be selected to form part of the next population of bees.

Steps 8 and 9 are optional. Linear shrinking is activated when there is no improvement of fitness value (shrinking procedure trigger is related to search space and problem type) and the abandonment procedure is triggered when trapped in local optima.

In step 10, the remaining unselected scout bees are sent randomly for global search to re-explore for any new potential solutions. The update of the fuzzy logic greedy selection system with the new fitnesses and ranks of the population is implemented in step 11.

Step 12 is the end of the loop. The search stops when the optimal solution of a problem is found or the stopping criterion is met, otherwise repeat last steps.

1. Initialise a population of (ns) scouts with random solutions.

2. Evaluate fitness of the population.

3. Form fuzzy greedy system with initial values of fitness and rank.

4. While (stopping criterion not met).

//Forming new population.

5. Select patches for neighbourhood search (each patch has its own memory to store its patch size).

6. Recruit bees for selected patches (more bees for best patches) and evaluate their fitness.

7. Select the fittest bee from each patch.

8. Shrink the patch size when it is needed (dividing the patch size by 2).

9. Abandon when trapped in a local peak.

10. Assign remaining bees to search randomly and evaluate their fitness.

11. Update fuzzy greedy system parameters.

12. End While.

Figure 3.7 Pseudo code of the enhanced Bees Algorithm

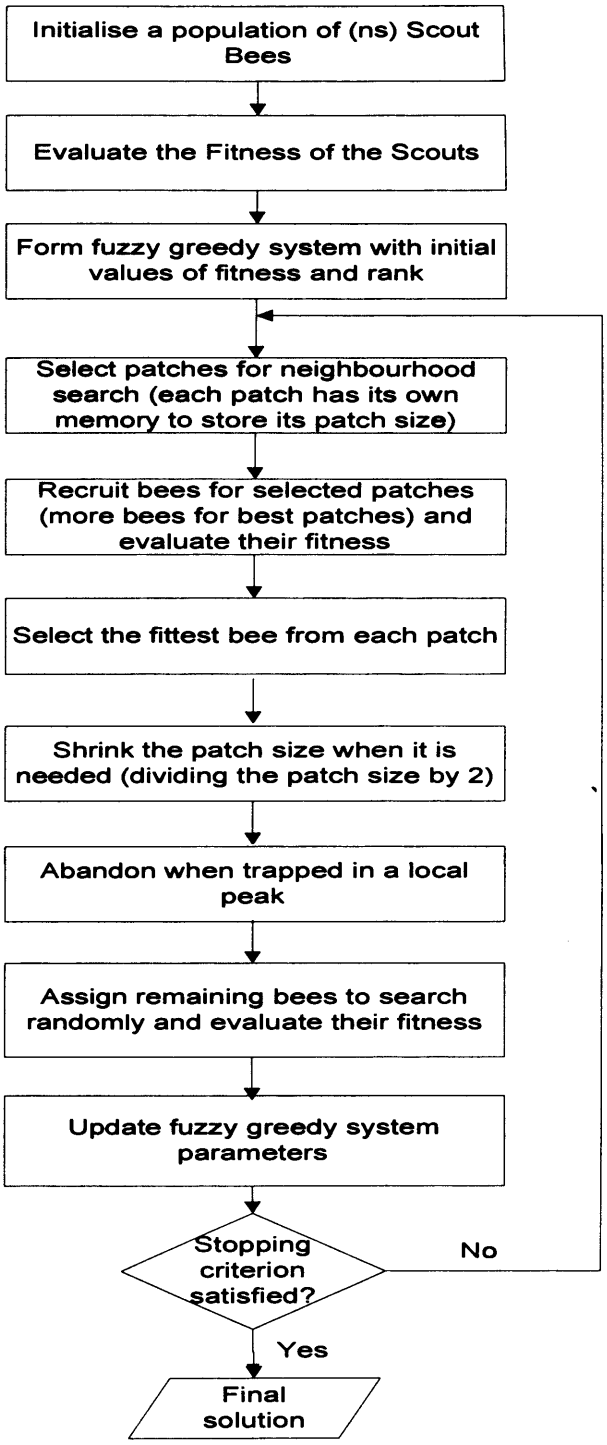


Figure 3.8 The flowchart of the enhanced Bees Algorithm

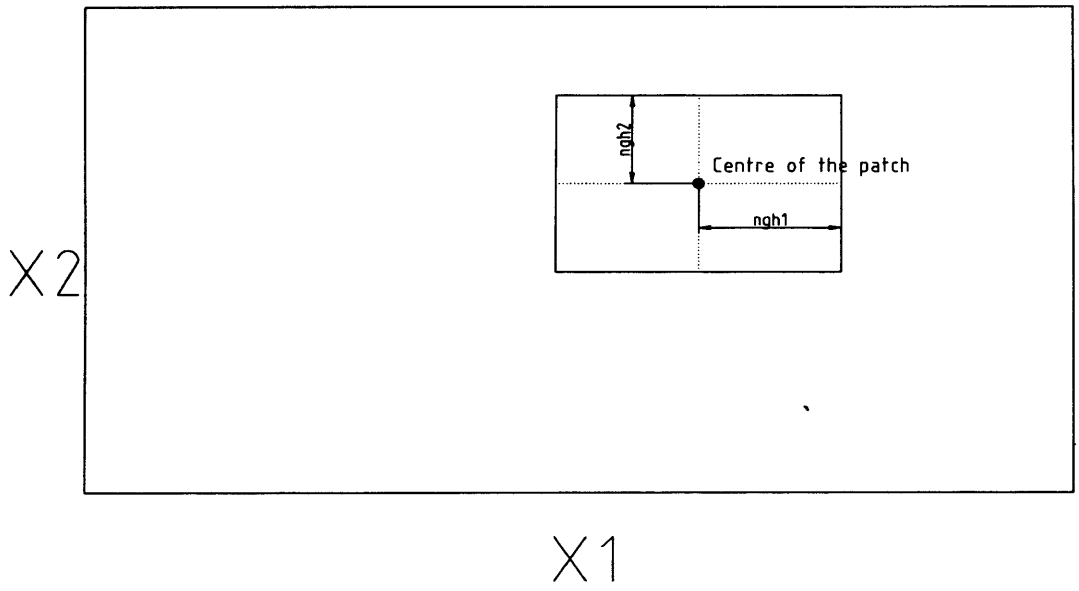


Figure 3.9 A local patch in a search space

Abandon patches in the Bees Algorithm is similar to a random-restart hill climbing (Russell and Norvig 2004). A random-restart hill climbing algorithm is supervised by an outer monitoring procedure which runs a hill climbing algorithm with a new initial state each time it stops improving. The best-found state is stored aside to be replaced with a better result from a successor run. This idea prevents the algorithm from becoming trapped in local optima.

Ghanbarzadeh (Ghanbarzadeh 2007) in his PhD thesis describes the abandonment method. It is used when the Bees Algorithm is trapped in local optima and involves eliminating the trapped patches from the candidate list.

3.4. Experiments

The enhanced Bees Algorithm has been applied to the eight benchmark functions given in (Mathur et al. 2000; Pham et al. 2006b) and the results compared with those obtained using other optimisation algorithms (Pham et al. 2006b). The test functions and their optima are shown in Table 3-1 and Table 3-2. The benchmark suite consists of 8 functions.

Table 3-3 and Table 3-4 show the results of applying several optimisation algorithms to the same benchmark suite. They also show the average number of evaluations needed to obtain the required optimal values using the enhanced Bees Algorithm. Table 3-5 and Table 3-6 show the used parameters of the enhanced

Bees Algorithm and the basic Bees Algorithm respectively for each one of the functions in the benchmark suite.

The tables present the results obtained by the enhanced Bees Algorithm and those by the deterministic Simplex method (SIMPSA), the stochastic simulated annealing optimisation procedure (NE SIMPSA), the Genetic Algorithm (GA), the Ant Colony System (ANTS) and the basic Bees Algorithm (Pham et al. 2006b).

The numbers of points visited shown are averages for 100 independent runs. The optimisation stopped when the difference between the maximum fitness obtained and the global optimum was less than 0.1% of the optimum value, or less than 0.001, whichever was smaller. In cases where the optimum was 0, the solution was accepted if it differed from the optimum by less than 0.001. If a solution is found that satisfies one of these conditions, the algorithm is said to have succeeded in finding the optimum.

No	Function Name	Interval	Function	Global Optimum
1	De Jong	[-2.048, 2.048]	$\max F = (3905.93) - 100(x_1^2 - x_2)^2 - (1 - x_1)^2$	X(1,1) F=3905.93
2	Goldstein & Price	[-2, 2]	$\min F = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $X[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	X(0,-1) F=3
3	Branin	[-5, 10]	$\min F = a(x_2 - b x_1^2 + c x_1 - d)^2 + e(1-f)\cos(x_1) + e$ $a = 1, b = \frac{5.1}{4}\left(\frac{7}{22}\right)^2, c = \frac{5}{22} X 7, d = 6, e = 10, f = \frac{1}{8} X \frac{7}{22}$	X(-22/7,12.275) X(22/7,2.275) X(66/7,2.475) F=0.3977272
4	Martin & Gaddy	[0, 10]	$\min F = (x_1 - x_2)^2 + ((x_1 + x_2 - 10)/3)^2$	X(5,5) F=0

Table 3-1 Test Functions

No	Function Name	Interval	Function	Global Optimum
5a	Rosenbrock (2D) (a)	[-1.2, 1.2]	$\min F = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	X(1,1) F=0
5b	Rosenbrock (2D) (b)	[-10, 10]	$\min F = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	X(1,1) F=0
6	Rosenbrock (4D)	[-1.2, 1.2]	$\min F = \sum_{i=1}^{3} \{100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2\}$	X(1,1,1,1) F=0
7	Hypersphere	[-5.12, 5.12]	$\min F = \sum_{i=1}^{6} x_i^2$	X(0,0,0,0,0,0) F=0
8	Griewank	[-512, 512]	$\min F = \frac{1}{4000} \cdot \left(\sum_{i=1}^{10} (x_i - 100)^2 \right) - \left(\prod_{i=1}^{10} \cos\left(\frac{x_i - 100}{\sqrt{i}} \right) \right) + 1$	X($\vec{1}$00) F=0

Table 3-2 Test Functions

funct. no.	SIMPSA		NE-SIMPSA		GA	
	succ %	mean no. of evaluations	succ %	mean no. of evaluations	succ %	mean no. of evaluations
1	****	****	****	****	100	10160
2	****	****	****	****	100	5662
3	****	****	****	****	100	7325
4	****	****	****	****	100	2844
5a	100	10780	100	4508	100	10212
5b	100	12500	100	5007	****	****
6	99	21177	94	3053	****	****
7	****	****	****	****	100	15468
8	****	****	****	****	100	200000

Table 3-3 Results for test functions

**** Algorithm did not converge

funct. no.	ANTS		Basic Bees Algorithm		Enhanced Bees Algorithm	
	succ %	mean no. of evaluations	succ %	mean no. of evaluations	succ %	mean no. of evaluations
1	100	6000	100	868	100	830
2	100	5330	100	999	100	212
3	100	1936	100	1657	100	184
4	100	1688	100	526	100	124
5a	100	6842	100	631	100	689
5b	100	7505	100	2306	100	1448
6	100	8471	100	28529	100	33367
7	100	22050	100	7113	100	526
8	100	50000	100	20998	100	8224

Table 3-4 Results for test functions

Function no	ns	nw
1	4	20
2	5	2
3	4	3
4	4	3
5a	3	20
5b	4	25
6	10	3
7	5	3
8	5	5

Table 3-5 The enhanced Bees Algorithm parameters

Function no	n	m	e	nsp	nep	ngh
1	10	3	1	2	4	0.1
2	20	3	1	1	13	0.1
3	30	5	1	2	3	0.5
4	20	3	1	1	10	0.5
5a	10	3	1	2	4	0.1
5b	6	3	1	1	4	0.5
6	20	6	1	5	8	0.1
7	8	3	1	1	2	0.3
8	10	3	2	4	7	5

Table 3-6 The basic Bees Algorithm parameters

(Pham et al. 2006b)

The first test function is De Jong's, for which the enhanced Bees Algorithm could find the optimum 7 times faster than ANTS, 11 times faster than GA and in almost the same time of the basic Bees Algorithm. The second function is Goldstein and Price's, for which the enhanced Bees Algorithm reached the optimum almost 25 times faster than ANTS, GA and 5 times faster than the basic Bees Algorithm.

The third function is Branin. The enhanced Bees Algorithm was 10 times faster than ANTS, 35 times faster than GA, and 9 times faster than the basic Bees Algorithm. Function 4 is Martin & Gaddy, for which the enhanced Bees Algorithm was 12 times faster than ANTS, 22 times faster than GA and 3 times faster than the basic Bees Algorithm.

Functions 5a and 5b are Rosenbrock's functions in two different intervals. In the first case, the enhanced Bees Algorithm delivered similar performance to the basic Bees Algorithm, was 10 times faster than ANTS and 15 times faster than GA. In the second case, the enhanced Bees Algorithm was 50% faster than the basic Bees Algorithm, 5 times faster than ANTS and 8 times faster than GA. In Rosenbrock function with 4 dimensions, the proposed algorithm was slightly slower than others; however it reached the optimum with success rate 100%.

Test function 7 is a Hyper Sphere model of six dimensions. The enhanced Bees Algorithm was 14 times faster than the basic Bees Algorithm, 40 times faster

than ANTS and roughly 30 times faster than GA. The eighth test function (Griewangk test function) is a ten-dimensional function. The enhanced Bees Algorithm could reach the optimum 2.5 times faster than the basic Bees Algorithm, 6 times faster than ANTS and 25 times faster than GA. The success rates were 100% for the functions used to test the performance of the enhanced Bees Algorithm.

From the tables it is clear that the enhanced Bees Algorithm could obtain the optimum for most of the functions in the benchmark suite used with a minimum number of tuneable parameters needed to run the algorithm.

3.5. Chapter summary

The Bees Algorithm is a population-based algorithm that mimics the natural food foraging behaviour of honey bees. The algorithm essentially involves both random exploration of the solution space and more focused exploitation of promising local search sites.

The enhanced Bees algorithm has been constructed from a combination of the Bees Algorithm as a core and a fuzzy logic system for greedy selection. The algorithm was applied to function optimisation as a maximisation problem. The work shown in this chapter proved that the fuzzy logic system employed reduced the number of parameters needed to run the Bees Algorithm.

CHAPTER 4. USING THE BEES ALGORITHM TO OPTIMISE A FUZZY LOGIC CONTROLLER

4.1. Preliminaries

The nonlinear characteristics of ill-defined and complex modern plants make classical controllers inadequate for such systems. However, using fuzzy sets and fuzzy logic principles has enabled researchers to better understand and hence control, complex systems that are difficult to model. These newly developed fuzzy logic controllers have given control systems a certain degree of intelligence.

A fuzzy logic controller or fuzzy controller can be considered a fuzzy rule-based controller which consists of input and output variables with membership functions, a set of (*IF* ... *THEN*) rules and an inference system. Designing fuzzy controllers involves deciding on appropriate values for the parameters of the fuzzy membership functions and constructing the rule base in order to achieve the required performance.

This problem can be solved by tuning the controller parameters using an optimisation technique to obtain the best possible solution according to a given

criterion or fitness function. Using the Bees Algorithm for optimising and adapting fuzzy logic systems is very convenient for the reason that fuzzy systems design depends on trial and error and the experience of the designer, which is similar to the heuristic characteristics of the Bees Algorithm.

4.2. ACROBOT

The ACROBOT simulated in this study is a planar robot consisting of two links with two joints. The structure of the robot is modelled on the body of a human gymnast balancing on a high bar, where the first joint of the robot represents the gymnast's fists gripping the bar, the first link his arms, head and torso, the second joint his hips, and the last link his legs and feet. One actuator is connected directly to the second joint while the first joint is left unpowered.

Controllers for the ACROBOT can be divided into two types: (i) up-swing controllers and (ii) balancing controllers.

The function of an up-swing controller is to move the ACROBOT from its stable state (in which it hangs vertically below the bar) to the inverted position (where the robot stands vertically upright, on its first joint, above the bar) by pumping energy from the second joint to the first joint. Methods such as using neural oscillators to eliminate the phase shift between the first and second joints (Matsuoka et al. 2006), fuzzy logic (Brown and Passino 1997; Smith et al. 1998), fuzzy neural network control (Zhao and Yi 2006) and partial feedback (Spong

1995) have been adopted to achieve up-swinging. The main task of the up-swing controller is to force the ACROBOT to enter the attraction basin of the inverted state with minimum velocity so as to enable the balancing controller to catch it and maintain it stably in the upright position (Brown and Passino 1997; Spong 1995; Wiklendt et al. 2008).

4.3. Dynamics model of ACROBOT

Figure 4.1 is a schematic diagram of the ACROBOT to be controlled. The robot is powered by a DC motor connected to the second joint using a belt and pulleys (Spong 1995). The description and values of the parameters shown are given in (Brown and Passino 1997).

A state-space model of the ACROBOT was obtained by linearising its dynamics around the inverted position $(q_1 = \pi / 2, \quad q_2 = 0, \quad \dot{q}_1 = \dot{q}_2 = 0)$ as

$$\dot{x} = Ax + B\tau \qquad \text{Equation 4.1}$$

$$y = Cx + D\tau \qquad \text{Equation 4.2}$$

where x, the state vector, is defined as:

$$x = [q_1 - \pi / 2 \quad q_2 \quad \dot{q}_1 \quad \dot{q}_2]^T$$

In Equation 4.1 and Equation 4.2, τ is the input torque applied to the actuator located at the second joint and $y=x$ is the output vector. With the robot parameters chosen as defined in (Brown and Passino 1997), A, B, C, and D are as follows:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 49.4782 & -5.5038 & 0 & 0 \\ -50.0109 & 66.2336 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ -23.9348 \\ 175.7326 \end{bmatrix}$$

$$C = I_{4\times 4}$$

$$D = 0_{4\times 1}$$

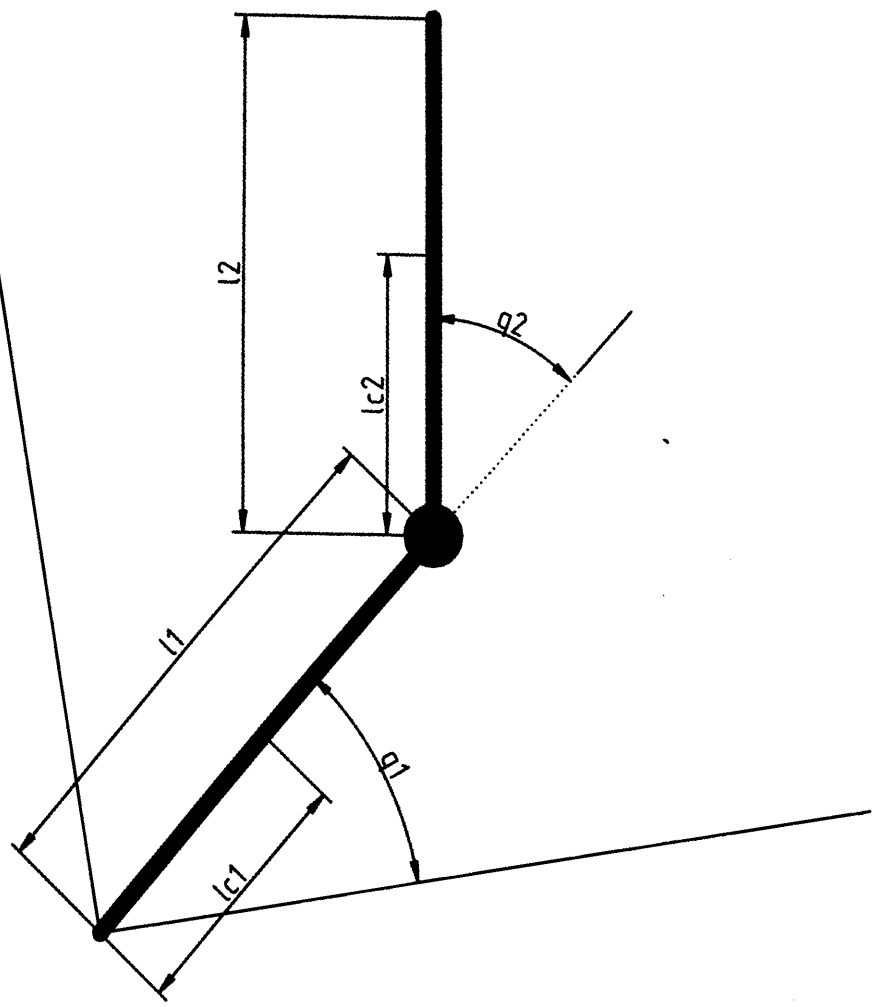


Figure 4.1 ACROBOT representation

4.4. The proposed controller

4.4.1. The LQR controller

The linearised ACROBOT model in Equation 4.1 and Equation 4.2 has been used to develop a linear quadratic regulator (*LQR*) (Anderson and Moore 1990) to maintain the robot balanced in a stable inverted position (Brown and Passino 1997; Spong 1995). Similar to the procedure followed in (Brown and Passino 1997), the controller gains were reproduced using a MATLAB standard *LQR* solution with the weight matrices Q and R chosen as follows:

$$Q=\begin{bmatrix} 1000 & -500 & 0 & 0 \\ -500 & 1000 & 0 & 0 \\ 0 & 0 & 1000 & -500 \\ 0 & 0 & -500 & 1000 \end{bmatrix}$$

$$R = [1000]$$

The obtained *LQR* is:

K_{LQR}= [-310.6372, -26.3246, -475231, 5.3165]

The *LQR* was employed to give the scaling gains of the fuzzy input and output variables needed for designing the alternative fuzzy logic controller that was subsequently tuned using the Bees Algorithm.

4.4.2. Fuzzy logic controller

The fuzzy controller consists of four input variables and one output variable. Each of the input variables has three membership functions defined in the universe of discourse [-1, 1] (see Figure 4.2). The output variable is composed of nine triangular membership functions with universe of discourse [-1, 1].

The LQR feedback gains were used to evaluate the scaling gains for the fuzzy logic controller. As in (Brown and Passino 1997), if the scaling gains for the input variables are denoted as $[g_0 \ \ g_1 \ g_2 \ g_3]$ and the scaling gain for the output variable is h, then $K_{LQR}=[g_0h \ \ g_1h \ \ g_2h \ \ g_3h]$.

Thus, a possible set of gains is:

$g_0 = 5.5555$, $g_1 = 0.4708$, $g_2 = 0.8500$, $g_3 = 0.0951$, and $h = 55.9147$. These are the same gains as those chosen for the controller reported in (Brown and Passino 1997).

The Mamdani model (Mamdani and Assilian 1975) is used as the basis of the proposed controller with the Max-Min method of inferencing and the Centroid method of defuzzification. The rule base consists of 81 (*IF ... THEN*) rules.

As in (Passino and Yurkovich 1998), Equation 4.3 is used to derive rules of the form :

If (q_1 is i) and (q_2 is j) and (dq_1 is k) and (dq_2 is l) then (action is m)

$$m = (i + j + k + l) \times \frac{2}{(N - 1)n} \qquad \text{Equation 4.3}$$

where m is the index of the membership function of the output action,

$i, j, k,$ and l are the indices of the input membership functions (i, j, k, l = 1, 2 or 3),

N is the number of input membership functions and n is the number of inputs.

Note that indices were subsequently converted into linguistic variables (for example, *NB, ZERO, PB* etc) for ease of reading the rules.

The following are two examples of rules from the rule base. The first rule is for i=j=k=l=1. The second rule is for i=3, j=2, k=1 and l=1.

Rule 1: *If (q1 is NB) and (q2 is NB) and (dq1 is NB) and (dq2 is NB) then (action is MF1)*

Rule 31: *If (q1 is PB) and (q2 is Zero) and (dq1 is NB) and (dq2 is NB) then (action is MF4)*

MATLAB and SIMULINK were used to implement the fuzzy controller and model the ACROBOT (see Figure 4.4).

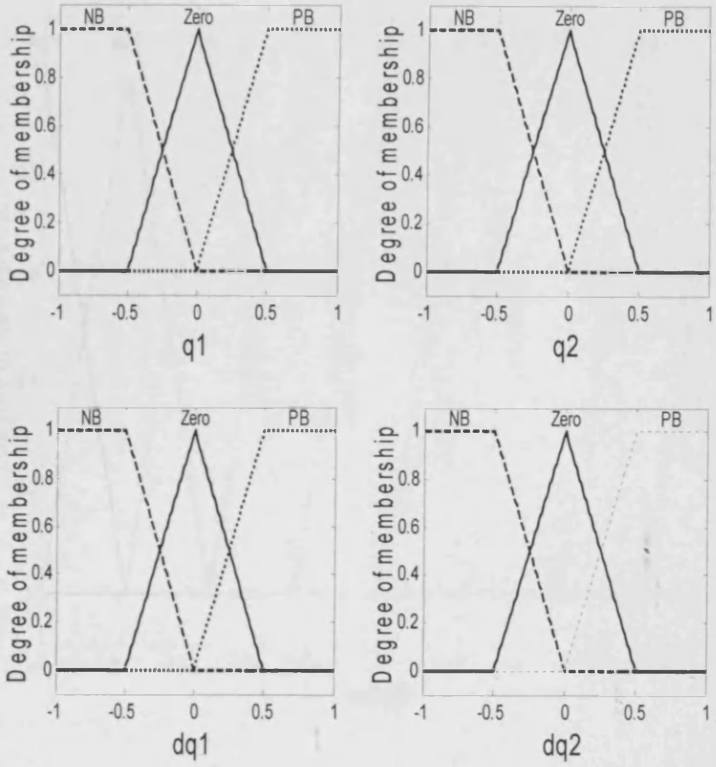


Figure 4.2 Input membership functions

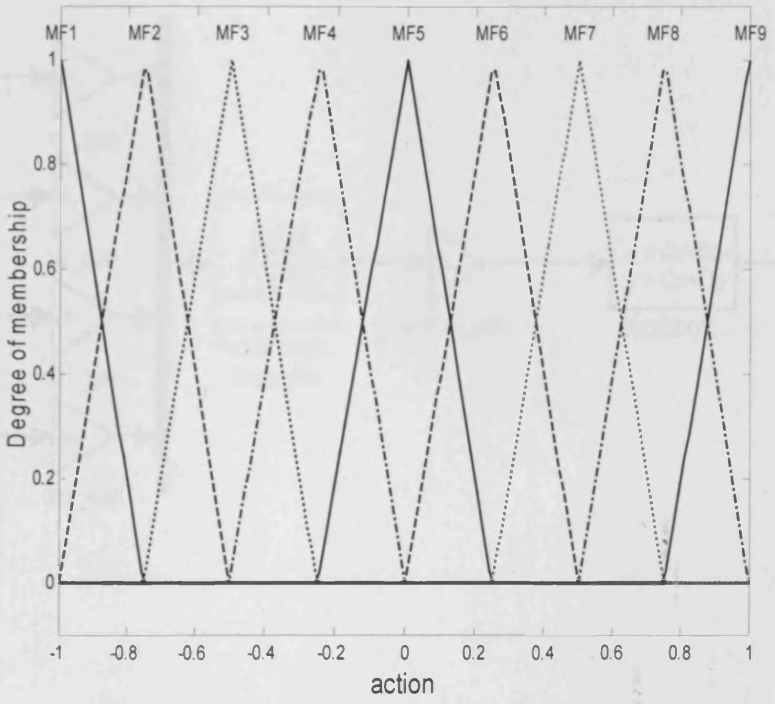


Figure 4.3 Output membership functions

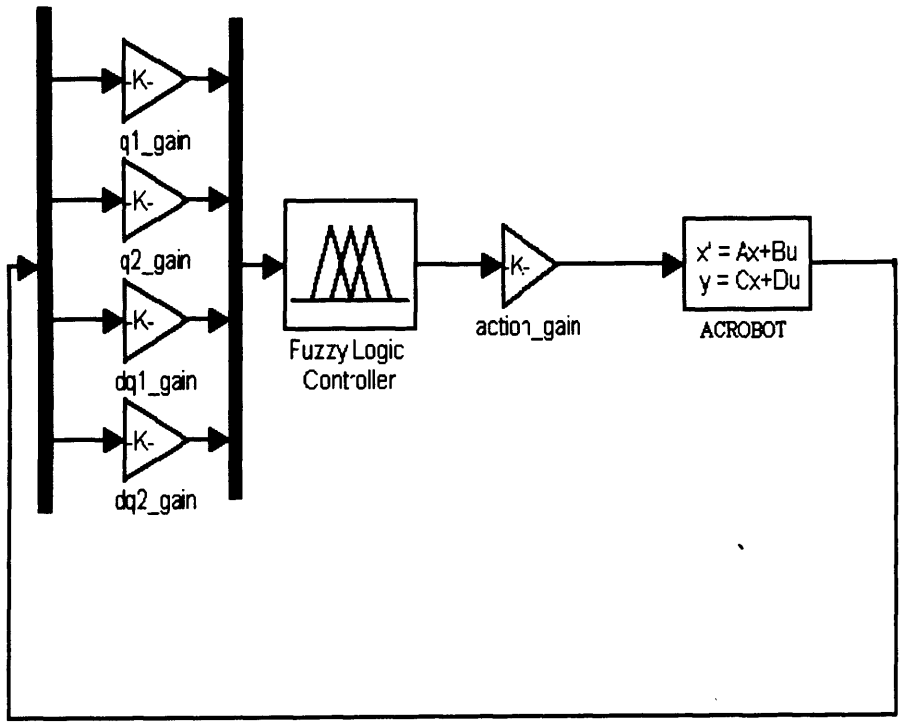


Figure 4.4 A SIMULINK representation of ACROBOT

4.5. Tuning of a fuzzy logic controller

4.5.1. Applying the Bees Algorithm

With the rule base fixed, the Bees Algorithm was used to tune the parameters of the input and output membership functions and the scaling gains for the input and output variables.

In theory, each bee was a vector comprising 60 real numbers. Five of those numbers were reserved for scaling gains, 11 for the parameters of each input variable membership function (three to represent a triangular function and four to represent a trapezoidal function) and 27 numbers to represent the triangular membership functions of the output variable.

However, due to symmetry and by appropriate design of the membership functions, a bee only needed to represent 12 numbers, one for each of the four input variables (*X1, X2, X3, X4*), three for the three output variables (*X5, X6, X7*), and five for the scaling gains (*X8, X9, X10, X11, X12*).

The search space was different for the numbers mentioned above. The search space for *X1, X2, X3* and *X4* to represent the membership functions of the input variables was [0, 1] and the spaces for the output variable membership functions *X5, X6* and *X7* were as follows:

X5: [0, 1]

X6: [*X5*, 1]

X7: [*X6*, 1]

The values of *X1, X2 ... X7* were used to construct the fuzzy controller. For example, the left-most input variable trapezoidal membership function for q1, [-1 -1 -*X1* 0], can be seen in Figure 4.5 which also shows the triangular function [-*X1* 0 *X1*] and the right-most (trapezoidal) function [0 *X1* 1 1].

The first input variable gain *X8* was in the range [0.2, 0.8], the second input gain *X9* belonged to [0.1, 2.5], the third input gain *X10* was in the range [0.1, 2.0] and the fourth input gain *X11* belonged to [0.01, 2.0]. The range of the output gain *X12* was [4.0, 99.0].

The ranges for the *X8, X9, X10, X11, X12* scaling gains were the same as those used in (Brown and Passino 1997).

The fuzzy controller derived from the *LQR* parameters given in section 3 was employed as a seed for the Bees Algorithm. The values of the parameters of the basic Bees Algorithm and enhanced Bees Algorithm are shown in Table 4-1 and Table 4-2 respectively.

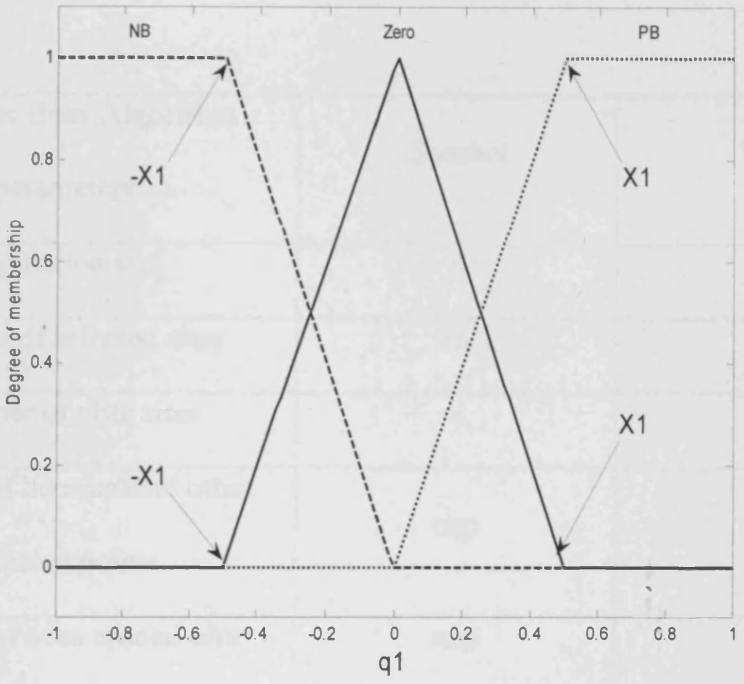


Figure 4.5 Construction of a membership function

The basic Bees Algorithm parameters	Symbol	Value
Population size	n	20
Number of selected sites	m	10
Number of elite sites	e	5
Number of bees around other selected points	nsp	3
Number of bees around elite	nep	10
Patch size	ngh	0.025

Table 4-1 The basic Bees Algorithm parameters

Enhanced Bees Algorithm parameters	Symbol	Value
Number of scouts	ns	3
maximum number of worker bees in each patch	nw	10

Table 4-2 The enhanced Bees Algorithm parameters

4.5.2. Fitness function

The fitness function was the same as that proposed in (Brown and Passino 1997). It is based on a weighted sum of parameters chosen to minimise the input torque τ, angular displacement $(q_1-\pi/2)$ of the first link away from the vertical inverted position and angular displacement q_2 away from the first link.

Let w be the vector of weights $[w_1 w_2 \ldots w_9]^T$ and S the vector of parameters $[S_1 S_2 \ldots S_9]^T$ The fitness function is given by:

$$f = \frac{1}{w^T S} \qquad \text{Equation 4.4}$$

f was calculated over a simulated control run of 10 seconds with the ACROBOT starting from rest to its inverted position.

The elements S_1 to S_9 are defined as follows:

S_1, S_2 and S_3 = mean value of $(q_1-\pi/2)$, q_2 and τ, respectively, over the time period 5-10 s, assuming the simulated control experiment started at time t=0 s.

S_4, S_5 and S_6 = normalised sum of squares $(q_1-\pi/2)^2$, $q_2^2$and τ^2, respectively, over the time period 0-5 s.

S_7, S_8 and S_9 = standard deviation of $(q_1-\pi/2)$, q_2 *and* τ, respectively, over the time period 5-10 s.

The weight vector used in (Brown and Passino 1997), w= [1 1 1 100 80 5 100 80 0.5], was also adopted as it had been found to reduce variations in q_1 and q_2 and to minimise the required input torque τ.

The Bees Algorithm was run for ten iterations and the parameters obtained at the end of the tenth iteration were taken as the tuned parameters of the control system.

4.6. Results

The simulation of the fuzzy controller was carried out under the MATLAB and SIMULINK environment with the fuzzy logic toolbox and the Runge-Kutta solver.

The time period for the simulation was 10 s. The state vector [$\pi/2$+0.04, -0.05, -0.2, 0.04] was used as the initial condition of the ACROBOT.

Table 4-3 shows the new values for the parameters of the fuzzy logic controller for a typical run of the Bees Algorithm (basic and enhanced).

Results before and after tuning are illustrated in Figure 4.6, Figure 4.7, Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11 respectively. Figure 4.6 shows the behaviour of the ACROBOT with the controller not tuned but it does illustrate the variations in the values of angles q_1 and q_2. Figure 4.7 shows the required control input to keep the ACROBOT balanced.

The behaviour of the ACROBOT as regards angles q_1, q_2 and control torque τ with a tuned controller by the Bees Algorithm (basic and enhanced) is shown in Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11.

From the simulation results, it is evident that the controller tuned using the Bees Algorithm (basic and enhanced) gave a smooth performance with fewer variations in the values of q_1, q_2 and smaller input control signals τ than in the case of the untuned controller. Hence, the Bees Algorithm is a useful tool for tuning fuzzy logic controllers to achieve better performance.

	Before tuning	After tuning	
		The basic Bees Algorithm	Enhanced Bees Algorithm
X1	0.5	0.5214	0.4763
X2	0.5	0.4832	0.4294
X3	0.5	0.5007	0.5148
X4	0.5	0.5051	0.4606
X5	0.25	0.2676	0.3083
X6	0.5	0.5451	0.562
X7	0.75	0.7174	0.6736
X8	5.5555	5.5691	5.5785
X9	0.4708	0.4494	0.4530
X10	0.8500	0.8737	0.8301
X11	0.0951	0.1038	0.0946
X12	55.9147	55.9472	55.9702

Table 4-3 Tuned parameters

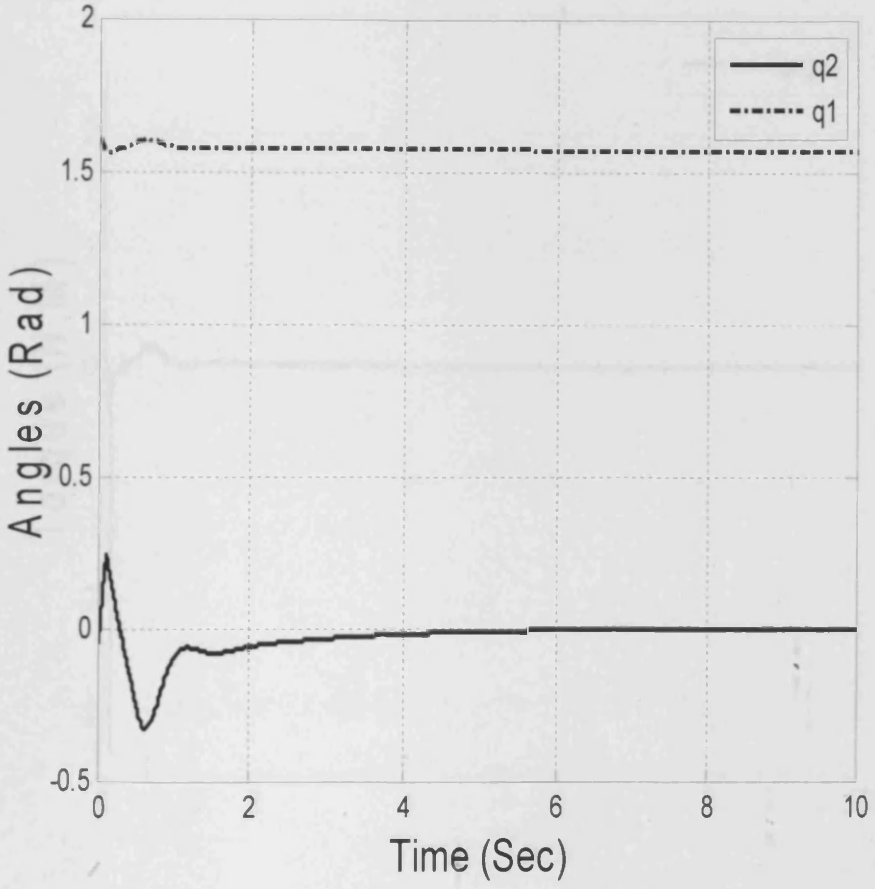


Figure 4.6 q_1, q_2 angles of balanced ACROBOT with the controller before tuning

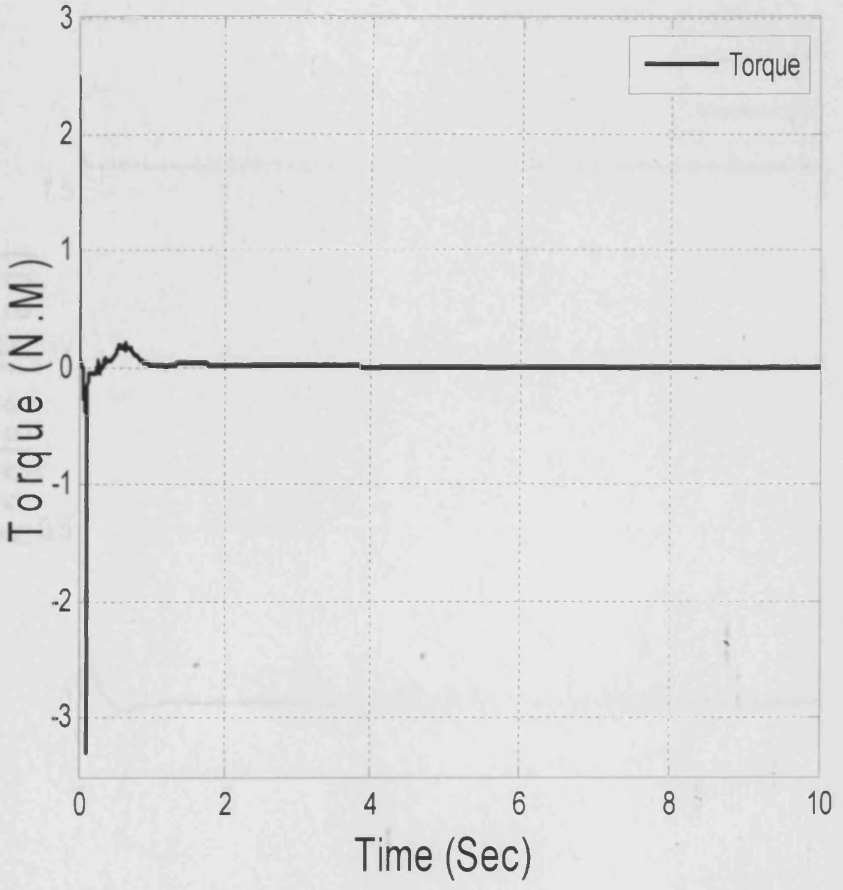


Figure 4.7 Control signal from the controller before tuning

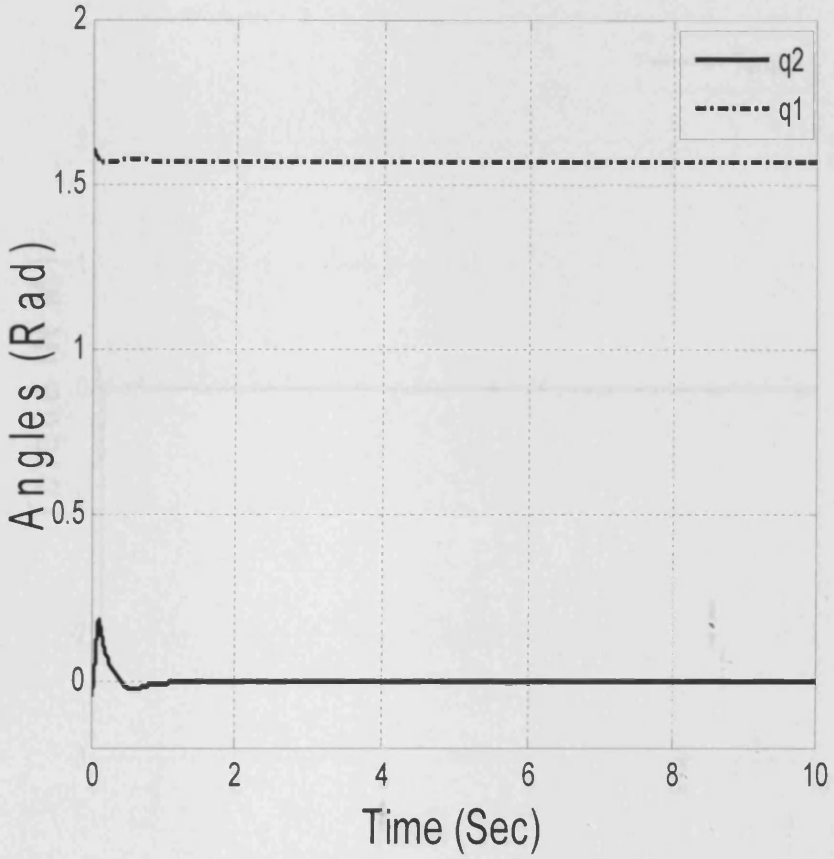


Figure 4.8 q_1, q_2 angles of balanced ACROBOT with tuned controller by the basic Bees Algorithm

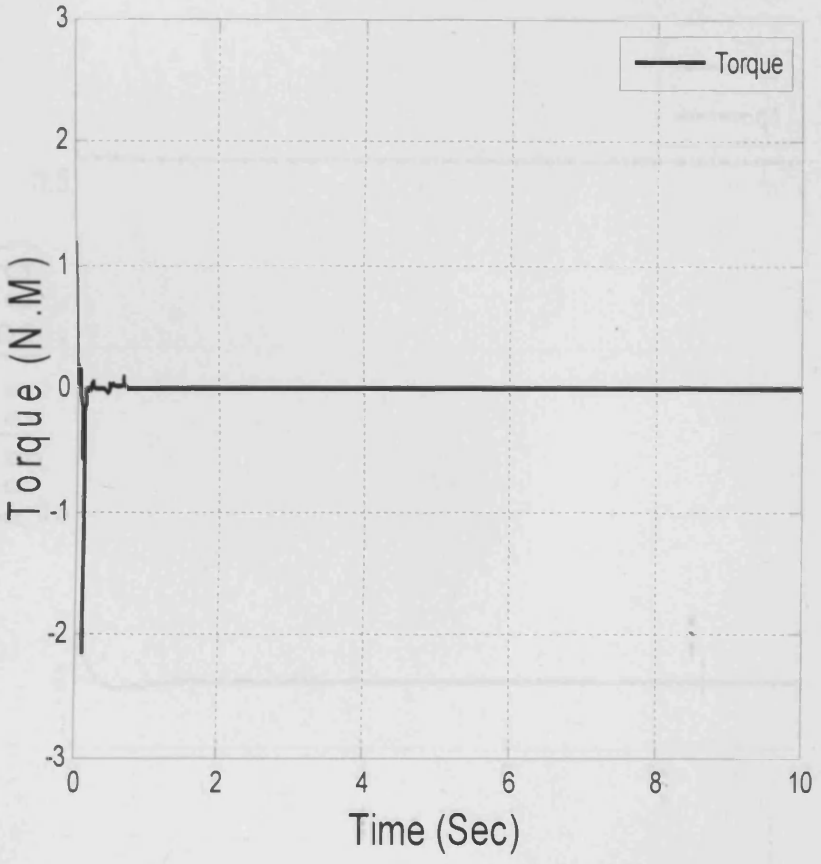


Figure 4.9 Tuned control signal by the basic Bees Algorithm

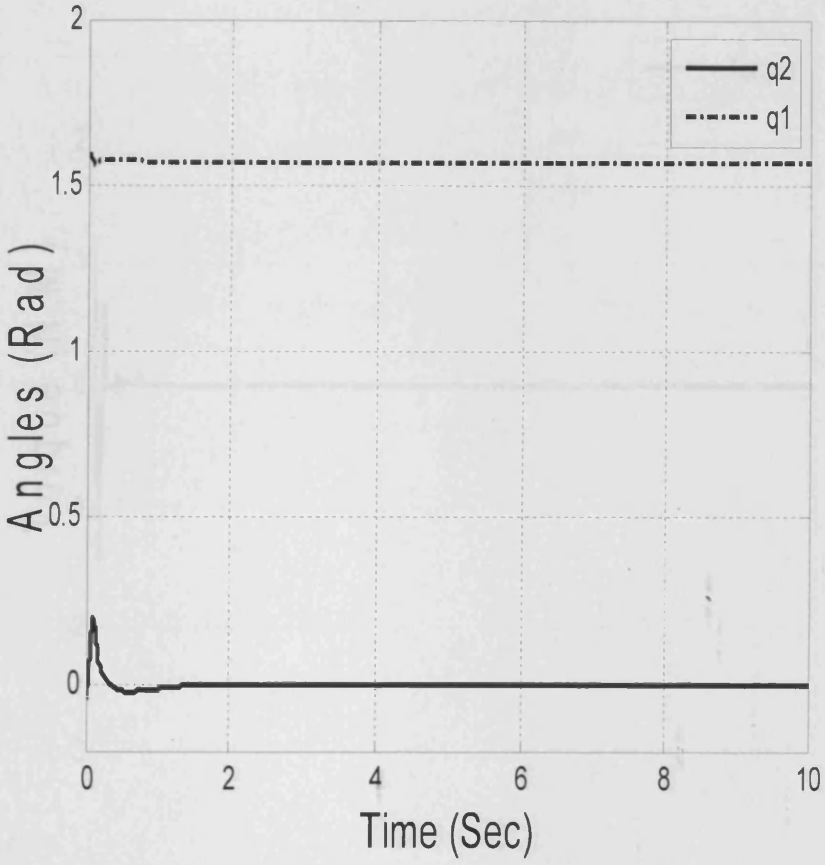


Figure 4.10 q_1, q_2 angles of balanced ACROBOT with tuned controller by enhanced Bees Algorithm

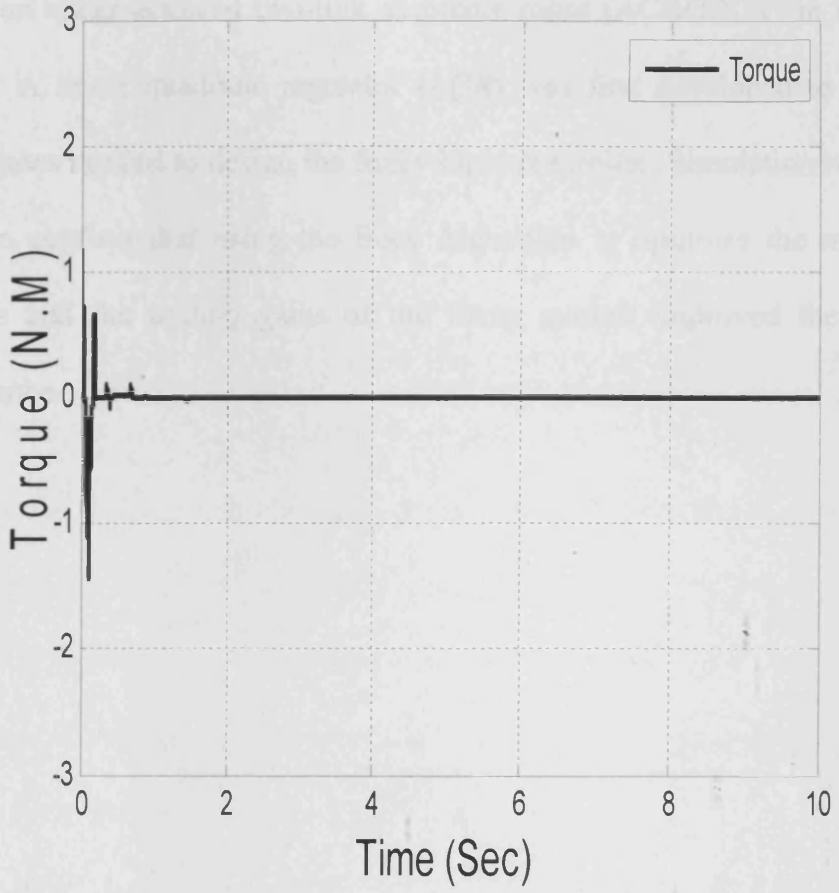


Figure 4.11 Tuned control signal by enhanced Bees Algorithm

4.7. Chapter summary

This chapter focused on using the Bees Algorithm in both its basic and enhanced forms to tune the parameters of a fuzzy logic controller developed to stabilise and balance an under-actuated two-link acrobatic robot (ACROBOT) in the upright position. A linear quadratic regulator (*LQR*) was first developed to obtain the scaling gains needed to design the fuzzy logic controller. Simulation results were shown to confirm that using the Bees Algorithm to optimise the membership functions and the scaling gains of the fuzzy system improved the controller performance.

CHAPTER 5. THE BEES ALGORITHM WITH KALMAN FILTERING AND ENHANCED FUZZY SELECTION

5.1. Preliminaries

This chapter focuses on combining a fast convergence gradient-based method with the Bees Algorithm and using the resulting algorithm to tune membership functions for a fuzzy logic system to minimise control errors. The proposed integration employs Kalman filtering as an alternative to random neighbourhood search to guide worker bees speedily towards the optima of local search sites. Fuzzy selection of local search sites is implemented to reduce the number of parameters needed to run the algorithm.

This chapter also presents the use of the Bees Algorithm with Kalman filtering, instead of the standard training algorithms, to train a Radial Basis Function (RBF) neural network. An enhanced fuzzy selection system has been developed to choose local search sites depending on the error and training accuracy of the RBF neural network.

Results of identification of wood defects with an RBF neural network trained using the Bees Algorithm with Kalman filtering and the conventional RBF procedure are shown and compared.

5.2. Integration of Kalman filtering with the Bees Algorithm

The position of a bee in the Bees Algorithm is a sample from the search space and, in a multi-dimensional function optimisation problem, represented as a vector of independent real numbers. In the standard Bees Algorithm, position updating in the local search part of the algorithm takes place in random jumps according to the following equation:

$$x_{new} = x_{old} + \alpha \cdot ngh \qquad \text{Equation 5.1}$$

where $\alpha \in uniform \quad (-1,+1)$

or $\alpha \in normal \quad (-1,+1)$

x_{new}, the new coordinates of a bee,

x_{old}, the most recent coordinates of a bee,

ngh, the radius of the local search patch.

Equation 5.1 is a simplified form of Equation 2.4, the recursive estimation or state update of the Kalman filter, which when linearised, can be written as:

$$\hat{x}_n = \hat{x}_{n-1} + K_n E_{n-1} \qquad \text{Equation 5.2}$$

where E_{n-1} is the Kalman estimation error and can be likened to the patch radius ngh of the Bees Algorithm.

Thus, the Kalman filter equation for state update, Equation 5.2, can be used instead of Equation 5.1 to change the positions of worker bees in the exploitation stage. It is assumed that all bees have their own memories to store the most recent values of their Kalman filter parameters.

In addition to this replacement of random jumps with Kalman filter state updating, fuzzy greedy selection is also employed to choose local search sites and to recruit worker bees.

A flowchart of the Bees Algorithm with Kalman filtering and fuzzy site selection is presented in Figure 5.1.

As with the standard Bees Algorithm, the modified algorithm starts in step 1 with (ns) scout bees being placed uniformly randomly in the search space. The fitness of each site visited by the scout bees is evaluated (i.e., the differences between the target and the obtained results are calculated) in step 2.

In step 3, the sites visited by the scout bees are ranked. The best sites are selected for exploitation (local search) in step 4 and bees are recruited for those sites in

step 5. Site selection and bee recruitment are performed according to the fuzzy greedy procedure detailed in chapter 3. The procedure is called greedy because it favours those sites with high fitness values: the higher the fitness value, the higher the rank and the larger the numbers of bees recruited. Site selection and bee recruitment are implemented by applying fuzzy rules thus eliminating the need to set hard thresholds. In step 5, the fitness values of the points visited by the recruited bees are evaluated and the Kalman filter parameters (the filter gains) for those bees are updated.

Step 6 involves ranking the points visited at each site and selecting the point with the highest fitness value to compete for further exploitation in the next iteration.

The optional step 7 is invoked when the optimisation process is deemed to be trapped at a local peak, in which case the Kalman filter parameters for the associated bees are reset to their initial values, or when a fitness plateau is detected, which causes stopping of exploitation at that site and abandonment of the site for a new location in the search space.

In step 8, unused scout bees (i.e., those not already 'working' at the points selected in step 6) are again sent randomly to explore the search space looking for other potential solutions.

In step 9, the new sites found by the scout bees are ranked together with the points selected in step 6. The process is repeated from step 4 until a stopping criterion is met.

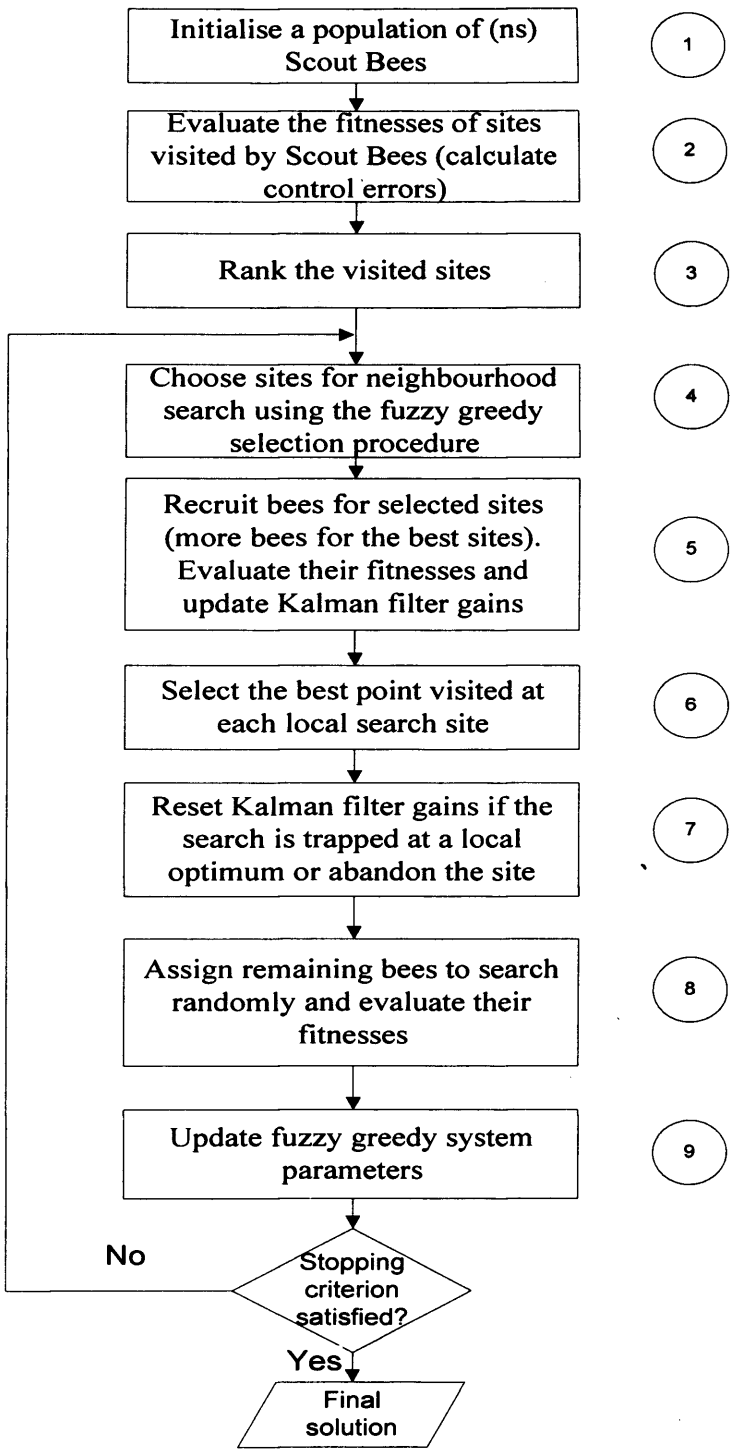


Figure 5.1 Flowchart of the Bees algorithm with Kalman filtering

An important advantage of using a Kalman filter to update the positions of bees is that the local search becomes adaptive. There is no need to pre-set the size of the local search area (i.e., the 'patch' size), nor to have a pre-determined schedule for shrinking the area: the extent of local search is controlled automatically for each bee by the Kalman filter gain K_n and estimation error E_n. Note that it is not critical to tune the filter precisely to reach good solutions. This is because the chances of finding them are high, given that the search for solutions proceeds from multiple starting positions.

5.3. Design of a fuzzy logic system

Consider the following equation which represents the dynamics of a vehicle powered by an engine and subjected to external drag and gravitational forces (Yen and Langari 1999):

$$m\frac{dv}{dt} = F_e(\theta) - F_d(v) - F_g \qquad \text{Equation 5.3}$$

where m is the vehicle mass, v the vehicle speed, F_e the engine force, F_d the drag force, F_g the gravity-induced force and θ the throttle position.

Equation 5.3 can be expressed in a more detailed form as:

$$\begin{aligned} F_e(\theta) &= F_i + \gamma \cdot \sqrt{\theta} \\ F_d(v) &= \alpha \cdot v^2 \cdot sign(v) \\ F_g &= m \cdot g \cdot \sin(grade) \end{aligned} \qquad \text{Equation 5.4}$$

The parameters in Equation 5.4 are defined in Table 5-1 which also gives the values adopted for them in this work.

A fuzzy system is designed to maintain a reference speed of the vehicle on a flat road with a sudden 10 degrees increase in the road grade at time = 0. The designed fuzzy system consists of two input variables and one output variable. Each of the input and output variables has five triangular membership functions.

Constant	value
Vehicle mass (m)	1000 kg
Drag coefficient (α)	4 N/(m/s)^2
Engine force coefficient (γ)	12,500 N
Engine idle force (F_i)	6,400 N
Engine time constant (τ_e)	0.1 to 1 second
Maximum throttle position (θ_{max})	30 to 60 degrees

Table 5-1 Vehicle constants

(Yen and Langari 1999)

The *ith* membership function of the *jth* input is represented by three parameters, namely, c_{ij}, b_{ij}^{-} and b_{ij}^{+}, which specify its centroid, lower half-width and upper half-width respectively. A membership degree for a given crisp input is defined by the following equation:

$$f_{ij}(x)=\begin{cases}1+(x-c_{ij})/b_{ij}^{-} & if \quad -b_{ij}^{-}\le(x-c_{ij})\le 0\\ 1-(x-c_{ij})/b_{ij}^{+} & if \quad 0\le(x-c_{ij})\le+b_{ij}^{+}\\ 0 & otherwise\end{cases} \qquad \text{Equation 5.5}$$

Correlation-product inference (Simon 2002a) is implemented with the Centroid defuzzification to compute the crisp output as follows:

$$Output=\frac{\sum_{j=1}^{n} m(\gamma_j)\gamma_j J_j}{\sum_{j=1}^{n} m(\gamma_j)J_j} \qquad \text{Equation 5.6}$$

where γ_j and J_j are the centroid and area of the *jth* output fuzzy membership function, and *n* is the number of output membership functions.

For the special case of two fuzzy inputs, the fuzzy output function $m(\gamma)$ is given as in:

$$m(\gamma)=\sum_{i,k} m_{ik}(\gamma) \qquad \text{Equation 5.7}$$

where $m_{ik}(\gamma)$ is the consequent fuzzy output function when input 1 is in class *i* and input 2 is in class *k*.

Table 5-2 shows the decision rules of the fuzzy system.

		ERROR				
		NL	*NS*	*Z*	*PS*	*PL*
ERROR CHANGE	*NL*	*NL*	*NL*	*NS*	*NS*	*NS*
	NS	*NL*	*NS*	*Z*	*Z*	*Z*
	Z	*NL*	*NS*	*Z*	*PS*	*PL*
	PS	*Z*	*Z*	*Z*	*PS*	*PL*
	PL	*PS*	*PS*	*PS*	*PL*	*PL*

Table 5-2 Fuzzy rules

(Simon 2002a)

5.4. Experimental results

With the rule base fixed and with an extended Kalman filter, the enhanced Bees Algorithm and the Bees Algorithm with Kalman filtering were used to tune the parameters of the input and output membership functions to achieve optimal results.

The parameters of the membership functions were assembled into a vector x

$$x = \left[b_{11}^{-} \quad b_{11}^{+} \quad c_{11} \quad \ldots b_{\mu 1}^{-} \quad b_{\mu 1}^{+} \quad c_{\mu 1} \quad .. \right] \qquad \text{Equation 5.8}$$

So the model of the fuzzy system is as in (Nian and Wunsch 2003; Simon 2002a).

$$x_{n+1} = x_n + w_n \qquad \text{Equation 5.9}$$

$$d_n = h(x_n) + v_n \qquad \text{Equation 5.10}$$

where $h(x_n)$ is the nonlinear mapping between the membership function parameters and the output,

w_n and v_n are artificially added noise processes,

d_n is the target output of the fuzzy system,

$h(\hat{x}_n)$ is the actual output.

The error function is defined as the reference speed minus the actual vehicle speed (Nian and Wunsch 2003; Simon 2002a). The simulation period is 15 s with 0.25 s sampling time and the target speed of the system is 40 m/s with a sudden 10 degrees increase in the road gradient at time = 0.

The behaviour of the vehicle optimised by the enhanced Bees Algorithm, extended Kalman filter and the integrated algorithm is shown in Figure 5.3, Figure 5.4 and Figure 5.5, while Figure 5.2 depicts the behaviour before optimisation. Table 5-3 and Table 5-4 show the parameters needed to run the enhanced Bees algorithm and the Bees Algorithm with Kalman filtering, respectively, where $\boldsymbol{I}$ is the identity matrix and 45 is the number of membership function parameters.

The extended Kalman filter was run for 100 iterations with initial parameters P=1e18*$\boldsymbol{I}_{45}$, Q=4000*$\boldsymbol{I}_{45}$ and R=1e-8.

It was found after 20 iterations that the integrated algorithm gave better results than those of the enhanced Bees Algorithm and the extended Kalman Filter either on their own.

Enhanced Bees Algorithm parameters	Symbol	Value
Number of scouts	ns	10
Maximum number of worker bees in each patch	nw	5

Table 5-3 Parameters of the enhanced Bees algorithm

The integrated algorithm parameters	Symbol	Value
Number of scouts	ns	10
Maximum number of worker bees in each patch	nw	5
covariance matrices of Kalman filter	$P=Q$ R	$10*I_{45}$ 10

Table 5-4 Parameters of the proposed algorithm

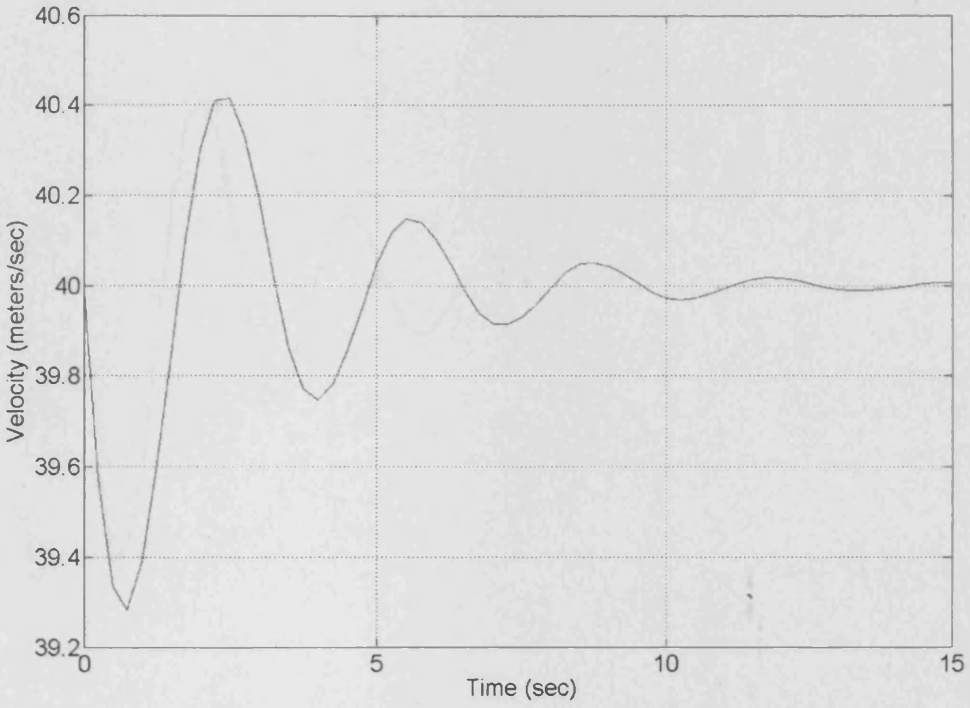


Figure 5.2 Velocity of the vehicle without optimisation

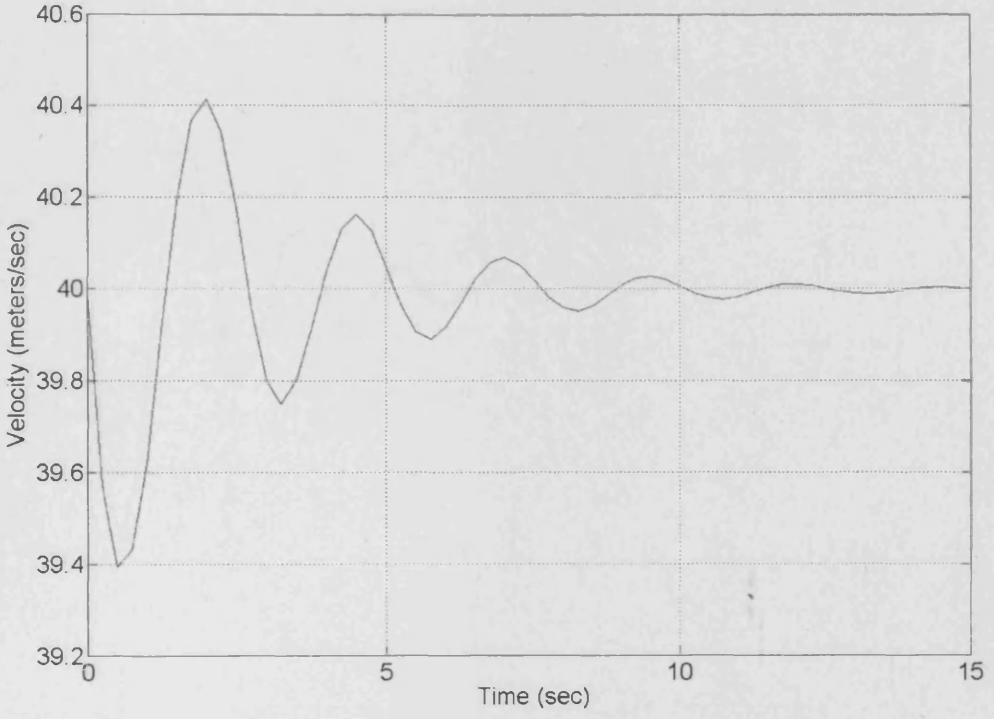


Figure 5.3 Velocity of the vehicle after optimisation by the enhanced Bees Algorithm

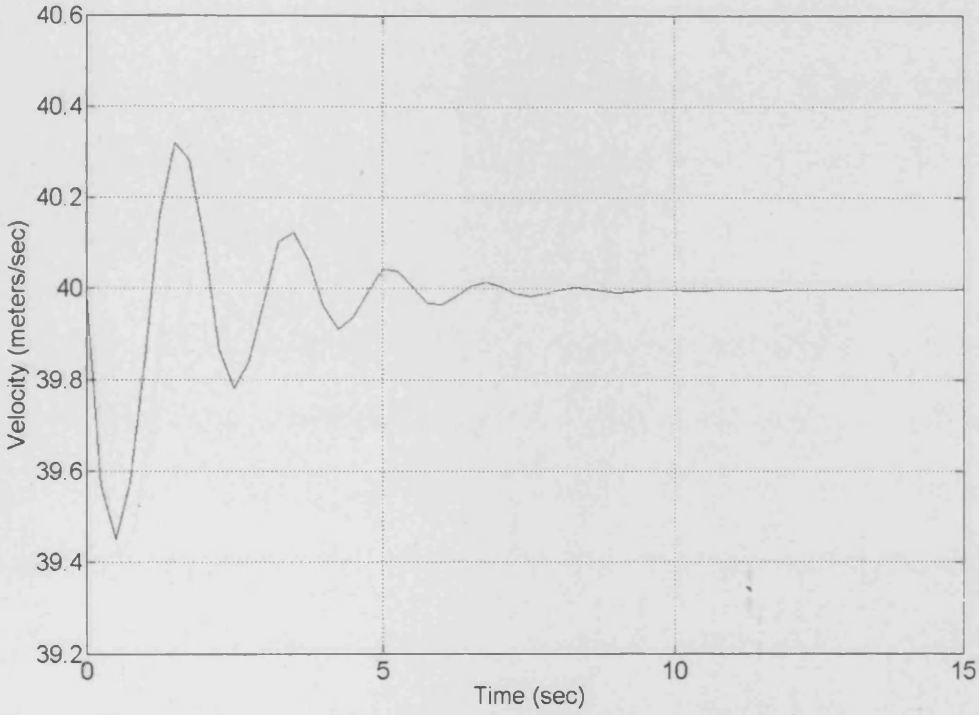


Figure 5.4 Velocity of the vehicle after optimisation by extended Kalman filter

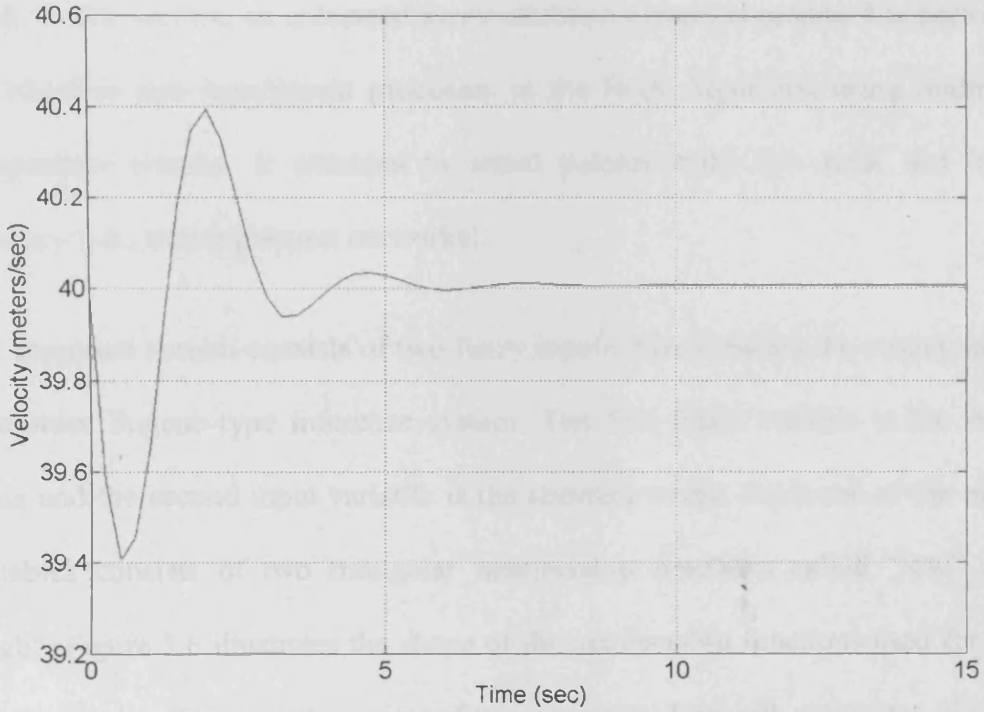


Figure 5.5 Velocity of the vehicle after optimisation by the integrated algorithm

5.5. Enhanced fuzzy selection

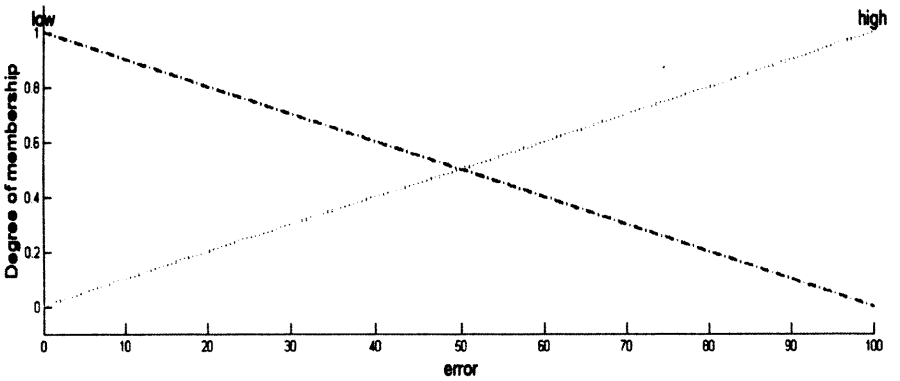
In Chapter 3, a fuzzy greedy selection system was constructed to choose local search sites and to decide on the number of recruited workers for each selected patch. In this section, an enhanced fuzzy selection system is proposed to perform the selection and recruitment processes in the Bees Algorithm using multiple independent criteria. It attempts to select patches with low error and high accuracy (i.e., training neural networks).

The proposed system consists of two fuzzy inputs, two constants for output and a zero-order Sugeno-type inference system. The first input variable is the error value and the second input variable is the accuracy value. Each one of the input variables consists of two triangular membership functions called "low" and "high". Figure 5.6 illustrates the shape of the membership functions used for the input variables. The output consists of two constants: low with value zero (0) and high with value (nw), which is the maximum number of worker bees per patch.

The initial universe of discourse for the inference system used comes from the maximum and the minimum values for error of the randomly visited patches and their training accuracy. The universe of discourse for the fuzzy system is updated at the end of each loop after sorting the candidate list. The repeated update makes the selection procedure dynamic.

The selection is performed using fuzzy rules shown in Figure 5.7. The structure of the rules gives the system its multicriteria selection behaviour, since recruitment depends on two independent terms.

The output is rounded to give the total number of worker bees in a selected patch. In this type of selection, there is no need to sort local search sites in a candidate list as ranking does not play any role in the fuzzy multicriteria selection.



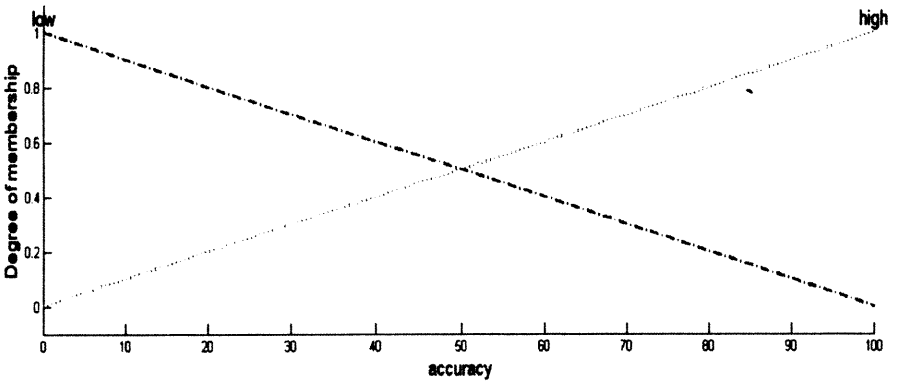


Figure 5.6 Input membership functions for enhanced fuzzy selection

1. If (error is high) and (training accuracy is high) then (recruitment is low)

2. If (error is high) and (training accuracy is low) then (recruitment is low)

3. If (error is low) and (training accuracy is low) then (recruitment is low)

4. If (error is low) and (training accuracy is high) then (recruitment is high)

Figure 5.7 The rules of fuzzy enhanced selection

5.6. The Bees Algorithm with enhanced fuzzy selection

The flowchart of the proposed method is presented in Figure 5.8. As with the standard Bees Algorithm, the modified algorithm starts in step 1 with (ns) scout bees being placed uniformly randomly in the search space. The fitness of each site visited by the scout bees is evaluated (i.e., calculate error and training accuracy of a neural network) in step 2.

In step 3, an enhanced fuzzy system is formed and initialised with the initial values of error and training accuracy of the sites (patches) visited by scouts.

The best sites are selected for exploitation (local search) in step 4 and bees are recruited for those sites in step 5. The site selection and bee recruitment are performed using the enhanced fuzzy selection procedure and are conducted according to the error and training accuracy associated with each site (more bees for lower error and higher training accuracy). Site selection and bee recruitment are implemented smoothly by applying fuzzy rules. In step 5, the fitness values of the points visited by the recruited bees are evaluated and the Kalman filter parameters (the filter gains) for those bees are updated.

Step 6 involves ranking the points visited at each site and selecting the point with the highest fitness value (error and training accuracy) to compete for further exploitation in the next iteration.

The optional step 7 is involved when the optimisation process is deemed to be trapped at a local peak, in which case the Kalman filter parameters for the associated bees are changed (double the Kalman filter parameters), or when a fitness plateau is detected, which causes stopping of exploitation at that site and abandonment of the site for a new location in the search space.

In step 8, unused scout bees (i.e., those not already 'working' at the points selected in step 6) are again sent randomly to explore the search space looking for other potential solutions.

In step 9, the new sites found by the scout bees with the points selected in step 6 are used to update the enhanced fuzzy selection system. The process is repeated from step 4 until a stopping criterion is met.

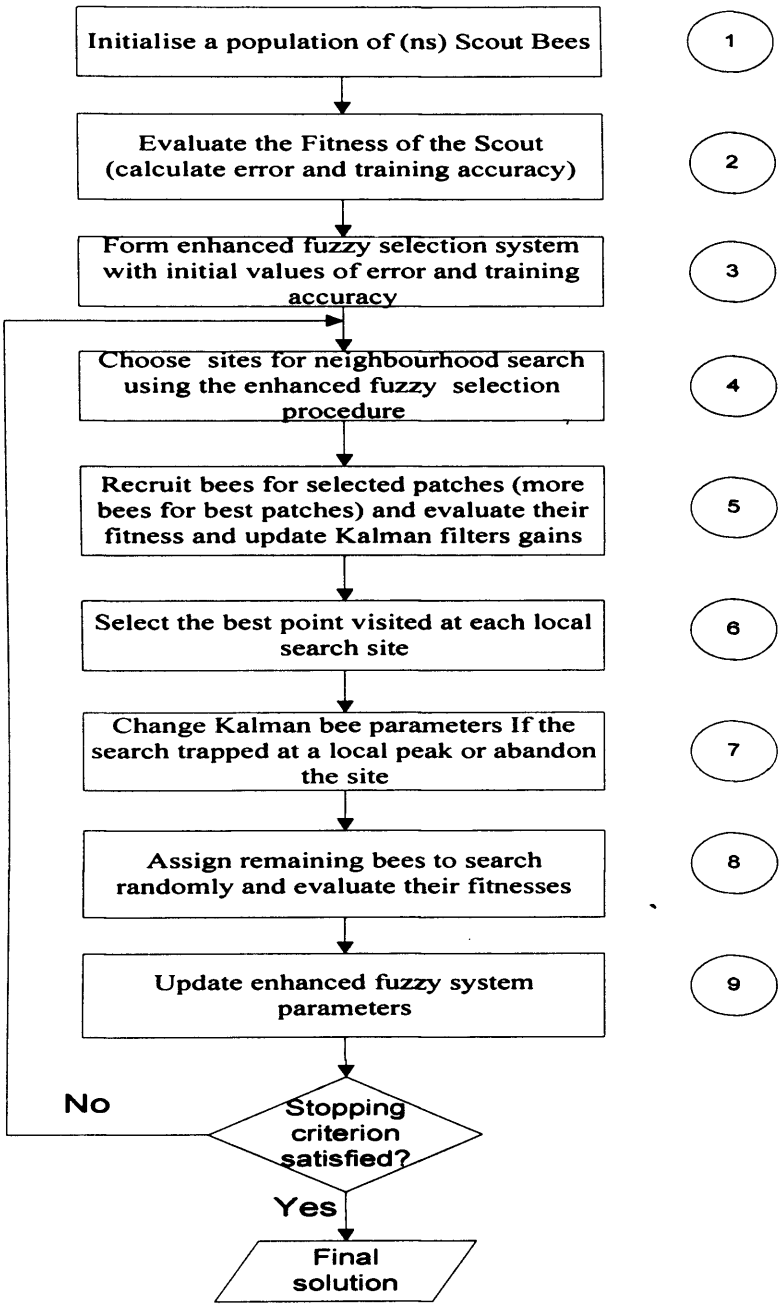


Figure 5.8 Flowchart of the proposed algorithm with enhanced fuzzy selection

5.7. Radial Basis Function (RBF)

A Radial Basis Function (RBF) is an Artificial Neural Network usually consisting of three layers of neurons.

In the RBF network shown in Figure 5.9 the input layer receives the input pattern which is the m-dimensional input x. The hidden layer (middle layer) consists of c neurons. Each of the c neurons in this layer applies an activation function which is a function of the Euclidean distance between the input and an m-dimensional prototype vector. Each hidden neuron contains its own prototype vector as a parameter. The output of each hidden neuron is then weighted and passed to the output layer. The outputs of the network consist of sums of the weighted hidden layer neurons (Simon 2002b).

The design of an RBF involves a decision on how many hidden neurons are to be included (integer value of c), the values of the (centres) prototypes (the values of the v vectors), the function to be used at the hidden units (function $g(\cdot)$) and the weights that will be applied between the hidden layer and the output layer (the values of the w weights) (Simon 2002b).

The function used in the hidden layer is of the general form (Chen et al. 1991; Simon 2002b) as in Equation 5.11 and Equation 5.12.

$$g(v) = [g_o(v)]^{1/(1-p)}$$ Equation 5.11

$$g_0(v) = norm(x - v)^2$$ Equation 5.12

where p is real number >0,

x input patterns,

v the centres of basis functions.

The output of the RBF network (y) is given by

$$y = K^T W^T$$ Equation 5.13

where K is the output of the hidden neurons and W is the weight matrix between the hidden layer and the output layer.

The elements of the weight matrix W and the elements of the prototypes v (centres of basis functions) form the state of a nonlinear system (see Equation 5.14) and the output of the RBF network forms the output of a nonlinear system.

$$x = \left[w_1^T \quad \ldots \quad w_n^T \quad v_1^T \quad \ldots \quad v_c^T \right]^T$$ Equation 5.14

The vector x consists of $(n(c+1)+mc)$ of the RBF parameters arranged in a linear array, where c is the number of hidden neurons, n the number of output neurons and m the number of input neurons.

$$x_{n+1} = f(x_n) + w_n \qquad \text{Equation 5.15}$$

$$y_n = h(x_n) + v_n \qquad \text{Equation 5.16}$$

where w_n and v_n are artificially added noise processes,

$f(.)$, the identity mapping,

y_n, the target output of the RBF network,

$h(x_n)$, the actual output of the RBF network.

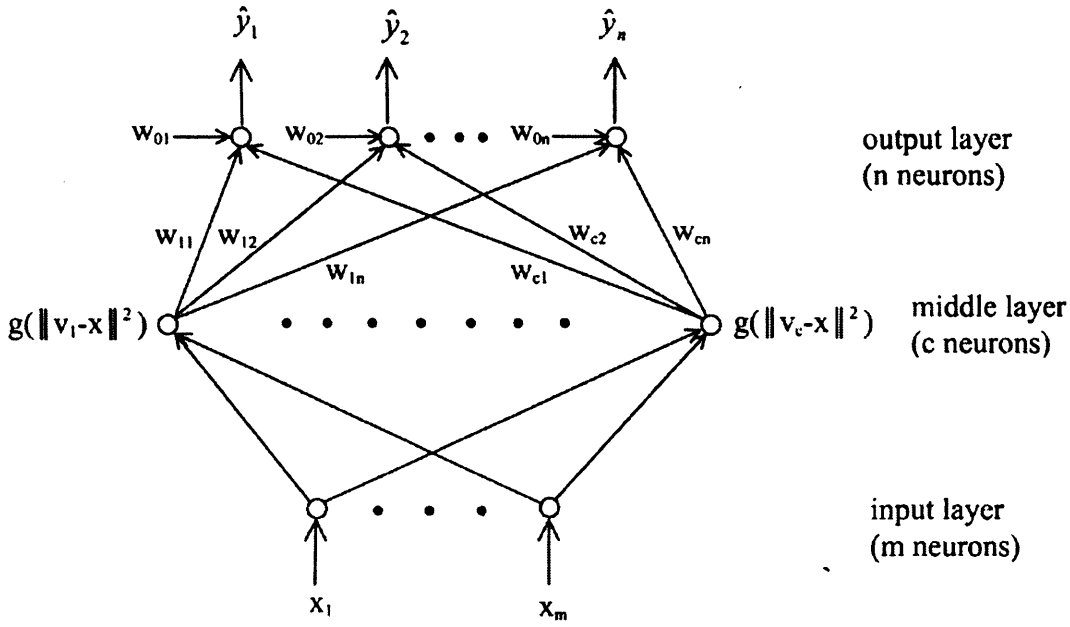


Figure 5.9 Structure of an RBF

(Simon 2002b)

5.8. Identification of wood defects

Automated Visual Inspection (AVI) systems for identifying wood defects using neural networks have been proposed by (Packianather and Drake 2005; Pham and Alcock 1996). Figure 5.10 shows a generic process of visual inspection for wood defects. Figure 5.11 illustrates twelve wood veneer defect and clear wood examples.

Wood defect data consists of 232 examples of defects and clear wood. Each example consists of 17 features. An RBF network is configured in three layers: an input layer with 17 neurons, a hidden layer with 51 neurons and an output layer with 13 neurons. The training set consists of 80% of wood defects data (185 in total) selected randomly and the remaining 20% (47 in total) formed the test set. Figure 5.12 summarises pattern classes and the number of examples used for training and testing.

The experiment was repeated ten times with ten iterations each. The number of scouts (ns) was five; maximum number of worker bees in each patch (nw) also was five and the covariance matrices of Kalman filter were $P=Q=10*I_{1543}$, where I is the identity matrix and 1543 is the size of vector x (see Equation 5.14), and $R=10*I_{2405}$, where 2405 is the number of output neurons multiplied by the number of patterns in the training set.

The average number of evaluations needed to obtain the results was 115 objective function calls instead of 100000 iterations with big population size (see Table 5-5) in the standard Bees Algorithm case (Ghanbarzadeh 2007).

Table 5-6 presents a comparison with conventional RBF training, MDC and the standard Bees Algorithm.

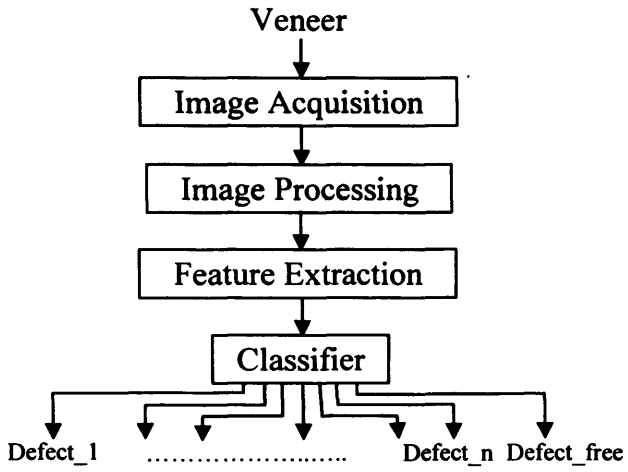


Figure 5.10 Generic Automated Visual Inspection system for wood defect identification

(Pham et al. 2006e)

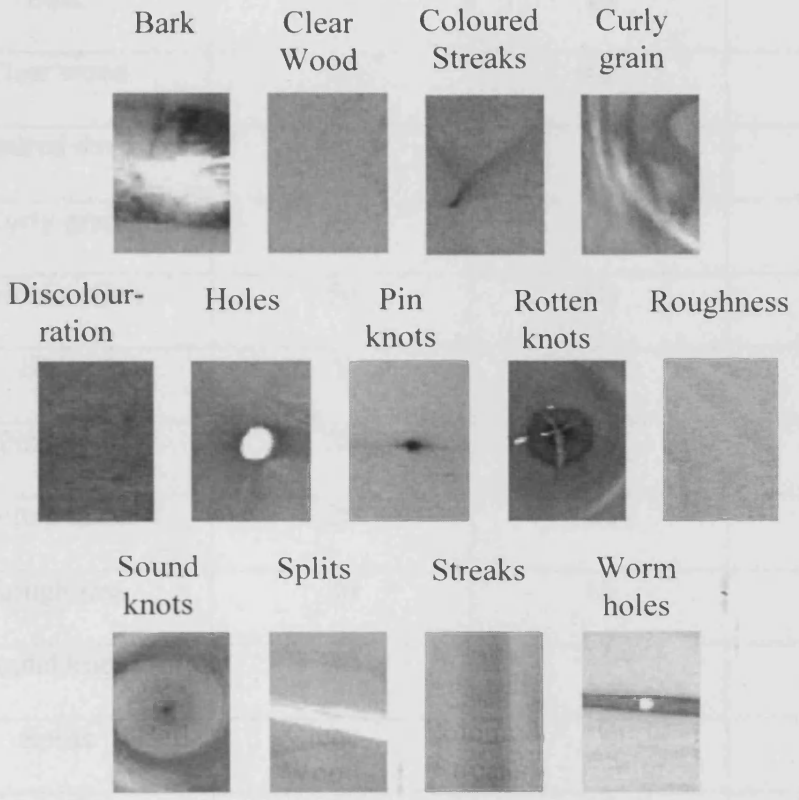


Figure 5.11 Wood veneer defect types

(Pham et al. 2006e)

Pattern Class	Total	Used for training	Used for Testing
Bark	20	16	4
Clear wood	20	16	4
Coloured streaks	20	16	4
Curly grain	16	13	3
Discoloration	20	16	4
Holes	8	6	2
Pin knots	20	16	4
Rotten knots	20	16	4
Roughness	20	16	4
Sound knots	20	16	4
Splits	20	16	4
Streaks	20	16	4
Wormholes	20	16	4
Total	232	185	47

Figure 5.12 Pattern classes and the number of examples used for training and testing

The Bees Algorithm parameters	Symbol	Value
Population	n	250
Number of selected sites	m	15
Number of elite sites out of m selected sites	e	3
Initial patch size	ngh	0.1
Number of bees for elite sites	nep	80
Number of bees for other selected sites	nsp	50

Table 5-5 The parameters of the standard Bees Algorithm

(Ghanbarzadeh 2007)

Pattern recognition	TRAINING ACCURACY	TEST ACCURACY
RBF (MATLAB) (Ghanbarzadeh 2007)	-	76.43%
MDC (Packianather and Drake 2005)	-	63.12%
RBF (The standard Bees Algorithm) (Ghanbarzadeh 2007)	86.9%	75.12%
RBF (The proposed algorithm)	91.08%	78.51%

Table 5-6 Comparison with conventional RBF training, MDC and the standard Bees Algorithm

5.9. Chapter summary

This chapter focused on merging Kalman filtering which is a gradient-based optimisation method with the Bees Algorithm. The Kalman filter enables rapid migration towards good solutions while premature convergence and sensitivity to initial positions are overcome by the swarm-based nature of exploration in the Bees Algorithm. A fuzzy greedy system was used to reduce the number of parameters needed to run the algorithm. The proposed algorithm was then used to optimise fuzzy membership functions of a dynamic system to produce minimal error.

An enhanced fuzzy selection system was developed and applied to the Bees Algorithm with Kalman filtering. The proposed method was used to train a Radial Basis Function (RBF) neural network for wood defect identification.

CHAPTER 6. CONCLUSION

This chapter gives a summary of the contributions and conclusions of this research. It also provides suggestions for future work.

6.1. Contributions

This research has introduced a number of developments to the Bees Algorithm to enhance it in terms of ease of use, robustness and speed.

The specific contributions were:

- Adopting a fuzzy logic system for the greedy selection of local search sites and applying the proposed method to function optimisation. Evaluating the Bees Algorithm in both its basic and enhanced forms applied to the problem of optimising a fuzzy logic controller for an under-actuated two-link acrobatic robot;
- Introducing Kalman filtering as a new way of performing local search in the Bees Algorithm and using the proposed method to tune membership functions for a fuzzy logic system;
- Employing an enhanced fuzzy selection system for the Bees Algorithm with Kalman filtering and applying the proposed algorithm to train a

Radial Basis Function (RBF) neural network for wood defect identification.

6.2. Conclusions

1. The proposed fuzzy greedy selection system and the proportional size of local patches to the search space intervals reduced the number of parameters needed to run the Bees Algorithm from six in the basic form to two in the enhanced form. The experimental results on continuous function optimisation showed the robustness of the new algorithm, with 100% success rate in all cases.

2. The application of the Bees Algorithm to the optimisation of the parameters of the acrobatic robot controller gave the robot a smooth performance and confirmed the superiority of the new algorithm compared to the basic version.

3. The combination of the Bees Algorithm with Kalman filtering for fuzzy membership functions tuning produced results that were better than those obtained using either on their own.

4. The use of enhanced fuzzy selection of local search sites eliminated the need to rank the search sites and gave the RBF neural classifier better training and test results than those of conventional training methods and the standard Bees Algorithm. The proposed method also reduced the number of evaluations needed to train the neural network.

6.3. Future work

This section discusses some of the ways in which the methods and algorithms developed in this thesis could be enhanced.

1. The enhanced Bees Algorithm developed in this work employs a fuzzy logic system that significantly reduces the number of the parameters of the algorithm. An area for further research is the investigation of the effect of using different types of membership functions with different parameters for the fuzzy system. It may also be possible to enhance the search process through the use of fuzzy neighbourhood search. Combinatorial optimisation could be considered an additional application area for the algorithm.

2. The Bees Algorithm was used to tune the parameters of a multi-input single-output (MISO) fuzzy logic controller. Further work could be carried out to apply the algorithm to multi-input multi-output (MIMO) fuzzy systems.

3. The Bees Algorithm was applied to optimise a fuzzy logic controller developed to stabilise and balance an acrobatic robot in the upright position. Future research also could focus on using the Bees Algorithm for the swinging-up phase with a view to developing a combined controller for both the swinging-up and balancing of the robot.

4. Kalman filtering was employed only as a new way of performing local search in the Bees Algorithm. Further work could be carried out to increase the efficiency and effectiveness of the Bees Algorithm by implementing the predictive attribute of the Kalman filter to control the whole flow of the search process in the algorithm (including local search site shrinking and abandonment).

5. The enhanced fuzzy selection system was developed by using multiple independent criteria. Another promising direction for research is to apply the enhanced fuzzy selection system to choose local search sites in multiobjective optimisation problems.

REFERENCES

Abraham, A., Liu, H., Zhang, W. and Chang, T.-G. 2006. Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm. In: *10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, KES 2006*. Bournemouth, UK: Springer. pp. 500-507.

Anderson, B. D. O. and Moore, J. B. 1990. *Optimal control: linear quadratic methods*. Upper Saddle River, NJ, USA: Prentice-Hall, p. 380.

Awadalla, M. H. A. 2005. *Adaptive co-operative mobile robots.* PhD thesis, Cardiff University.

Bagis, A. 2003. Determining fuzzy membership functions with tabu search-an application to control. *Fuzzy Sets and Systems* 139(1), pp. 209-225.

Bahamish, H. A. A., Abdullah, R. and Abdul Salam, R. A. 2008. Protein Conformational Search Using Bees Algorithm. In: *2nd IEEE Asia International Conference on Modeling & Simulation, AICMS 08*. Kuala Lumpur, Malaysia: IEEE Computer Society. pp. 911-916.

Baker, B. M. and Ayechew, M. A. 2003. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research* 30(5), pp. 787-800.

Bar-Yam, Y. 2003. *DYNAMICS OF COMPLEX SYSTEMS.* Boulder, Colorado: Westview Press, p. 864.

Bell, J. E. and McMullen, P. R. 2004. Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics* 18(1), pp. 41-48.

Bezdek, J. C. 1981. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Norwell, MA, USA: Kluwer Academic Publishers, p. 256.

Bohachevsky, I. O., Johnson, M. E. and Stein, M. L. 1986. Generalized simulated annealing for function optimization. *Technometrics* 28(3), pp. 209-217.

Bontoux, B. and Feillet, D. 2008. Ant colony optimization for the traveling purchaser problem. *Computers and Operations Research* 35(2), pp. 628-637.

Braun, H. 1990. On solving travelling salesman problems by genetic algorithms. In: *1st Workshop on Parallel Problem Solving from Nature.* Dortmund, Germany: Springer-Verlag. pp. 129 -133.

Brown, S. C. and Passino, K. M. 1997. Intelligent Control for an Acrobot. *Journal of Intelligent and Robotic Systems* 18, pp. 209-248.

Camazine, S., Deneubourg, J.-L., Franks, N., Sneyd, J., Theraula, G. and Bonabeau, E. 2003. *Self-Organization in Biological Systems: (Princeton Studies in Complexity).* Princeton: Princeton University Press, p. 560.

Castillo, P., Merelo, J., González, J., Rivas, V. and Romero, G. 1999. SA-prop: Optimization of multilayer perceptron parameters using simulated annealing. *Foundations and Tools for Neural Modeling*. pp. 661-670.

Chen, S., Cowan, C. F. N. and Grant, P. M. 1991. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks* 2(2), pp. 302-309.

Cheng, C.-B. and Mao, C.-P. 2007. A modified ant colony system for solving the travelling salesman problem with time windows. *Mathematical and Computer Modelling* 46(9-10), pp. 1225-1235.

Ciocoiu, I. B. 2002. RBF networks training using a dual extended Kalman filter. *Neurocomputing* 48(1-4), pp. 609-622.

Dadam, Y. 2002. *Fuzzy logic control of non-linear processes.* PhD thesis, Cardiff University.

Dadone, P. 2001. *Design Optimization of Fuzzy Logic Systems.* PhD thesis, Virginia Polytechnic Institute and State University.

Dorigo, M. and Blum, C. 2005. Ant colony optimization theory: a survey. *Theoretical Computer Science* 344(2-3), pp. 243-278.

Dorigo, M. and Gambardella, L. M. 1997. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation* 1(1), pp. 53-67.

Dorigo, M., Maniezzo, V. and Colorni, A. 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 26(1), pp. 29-41.

Eberhart, R. and Kennedy, J. 1995. A New Optimizer Using Particle Swarm Theory. In: *6th International Symposium on Micro Machine and Human Science.* Nagoya, Japan: IEEE press. pp. 39-43.

Efraim, T. and Louis, E. F. 1992. *Expert Systems and Applied Artificial Intelligence.* New Jersey, USA: Prentice Hall, p. 804.

Eickhoff, R., Kaulmann, T. and Ruckert, U. 2006. SIRENS: A Simple Reconfigurable Neural Hardware Structure for artificial neural network implementations. In: *International Joint Conference on Neural Networks, IJCNN '06.* Vancouver, BC, Canada: IEEE. pp. 2830-2837.

Engelbrecht, A. 2005. *Fundamentals of Computational Swarm Intelligence.* Hoboken, N.J: Wiley, p. 599.

Fahmy, A. A. 2005. *Neuro-fuzzy modelling and control of robotic manipulators.* PhD thesis, Cardiff University.

Feng, H.-M. 2005. Particle Swarm Optimization Learning Fuzzy Systems Design. In: *3rd International Conference on Information Technology and Applications ICITA*. Sydney, Australia: IEEE Computer Society. pp. 363-366.

Fort, J. C. 1988. Solving a combinatorial problem via self-organizing process: An application of the Kohonen algorithm to the traveling salesman problem. *Biological Cybernetics* 59(1), pp. 33-40.

Garcia-Naijera, A. and Brizuela, C. A. 2005. PCB Assembly: An Efficient Genetic Algorithm for Slot Assignment and Component Pick and Place Sequence Problems. In: *IEEE Congress on Evolutionary Computation*. Edinburgh, Scotland: IEEE Press. pp. 1485-1492.

Gendreau, M., Laporte, G. and Seguin, R. 1996. A Tabu Search Heuristic for the Vehicle Routing Problem with Stochastic Demands and Customers. *OPERATIONS RESEARCH* 44(3), pp. 469-477.

Ghanbarzadeh, A. 2007. *THE BEES ALGORITHM A Novel Optimisation Tool.* PhD thesis, Cardiff University.

Glover, F. 1989. Tabu Search--Part I. *INFORMS JOURNAL ON COMPUTING* 1(3), pp. 190-206.

Glover, F. 1990. Tabu Search--Part II. *INFORMS JOURNAL ON COMPUTING* 2(1), pp. 4-32.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Mass: Addison-Wesley, p. 372.

Gudise, V. G. and Venayagamoorthy, G. K. 2003. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In: *IEEE Swarm Intelligence Symposium*. Indianapolis, Indiana, USA: IEEE Press. pp. 110-117.

Guney, K. and Onay, M. 2007. Amplitude-only pattern nulling of linear antenna arrays with the use of bees algorithm. *Progress In Electromagnetics Research, PIER* 70, pp. 21–36.

Guney, K. and Onay, M. 2008. Bees algorithm for design of dual-beam linear antenna arrays with digital attenuators and digital phase shifters. *International Journal of RF and Microwave Computer-Aided Engineering* 18(4), pp. 337-347.

Guohua, W. and Yen, P. C. 1999. A fuzzy logic system for dynamic job shop scheduling. In: *IEEE International Conference on Systems, Man, and Cybernetics SMC '99*. Tokyo, Japan. pp. 546-551.

Hashiba, S. and Chang, T. C. 1992. Heuristic and Simulated Annealing Approaches to PCB Assembly Setup Reduction. In: *IFIP TC5 / WG5.3 Eight*

International PROLAMAT Conference on Human Aspects in Computer Integrated Manufacturing. Tokyo, Japan: North-Holland Publishing Co. pp. 769-777.

Haykin, S. S. 1999. *Neural networks : a comprehensive foundation*. 2nd ed. Upper Saddle River, N.J: Prentice Hall, p. 842.

Herrera, F., Lozano, M. and Verdegay, J. L. 1995. Tuning Fuzzy Logic Controllers by Genetic Algorithms. *International Journal of Approximate Reasoning* 3-4(12), pp. 299-315.

Ho, W. and Ji, P. 2005. A Genetic Algorithm to Optimise the Component Placement Process in PCB Assembly. *International Journal of Advanced Manufacturing Technology* 26(11), pp. 1397–1401.

Ho, W. and Ji, P. 2006. A Genetic Algorithm Approach to Optimising Component Placement and Retrieval Sequence for Chip Shooter Machines. *International Journal of Advanced Manufacturing Technology* 28(5), pp. 556–560.

Holland, A. 2003. Planning time restricted logistic tours with fuzzy logic. In: *3rd Conference of the European Society for Fuzzy Logic and Technology EUSFLAT*. Zittau, Germany: University of Applied Sciences at Zittau. pp. 329-334.

Holland, J. H. 1975. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Holland, J. H. 1992. *Adaptation in natural and artificial systems*. 1st ed. Cambridge, MA, USA: MIT Press, p. 211.

Jain, P. K. and Sharma, P. K. 2005. Solving job shop layout problem using ant colony optimization technique. In: *IEEE International Conference on Systems, Man and Cybernetics*. Waikoloa, Hawaii, USA: IEEE. pp. 288-292.

Jang, J.-S. R., Sun, C.-T. and Mizutani, E. 1997. *Neuro-Fuzzy and Soft Computing*. Upper Saddle River, NJ: Prentice Hall, p. 614.

Jones, K. O. and Bouffet, A. 2008. Comparison of bees algorithm, ant colony optimisation and particle swarm optimisation for PID controller tuning. In: *9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*. Gabrovo, Bulgaria: ACM.

Kalman, R. E. 1960. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME Journal of Basic Engineering* (82 (Series D)), pp. 35-45.

Kamrul Hasan, S. M., Sarker, R. and Cornforth, D. 2007. Hybrid Genetic Algorithm for Solving Job-Shop Scheduling Problem. In: *6th IEEE/ACIS*

International Conference on Computer and Information Science. Melbourne, Australia: IEEE Computer Society. pp. 519-524.

Karaboga, D. and Basturk, B. 2008. On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing* 8(1), pp. 687-697.

Kate, A. S. and Jatinder, N. D. G. 2001. Continuous Function Optimisation via Gradient Descent on a Neural Network Approximation Function. In: *6th International Work-Conference on Artificial and Natural Neural Networks: Connectionist Models of Neurons, Learning Processes and Artificial Intelligence-Part I*. Granada, Spain: Springer-Verlag. pp. 741-748.

Kennedy, J. 1997. The particle swarm: social adaptation of knowledge. In: *IEEE International Conference on Evolutionary Computation*. Indianapolis, IN, USA: IEEE. pp. 303-308.

Kennedy, J. and Eberhart, R. 1995. Particle swarm optimization. In: *IEEE International Conference on Neural Networks*. Perth, Australia: IEEE. pp. 1942-1948.

Khoo, L. P. and Loh, K. M. 2000. A Genetic Algorithms Enhanced Planning System for Surface Mount PCB Assembly. *International Journal of Advanced Manufacturing Technology* 16, pp. 289-297.

Khoo, L. P. and Ng, T. K. 1998. A Genetic Algorithm-based Planning System for PCB Component Placement. *International Journal of Production Economics* 54(3), pp. 321-332.

Kirkpatrick, S., Gelatt, C. D., Jr. and Vecchi, M. P. 1983. Optimization by Simulated Annealing. *Science* 220(4598), pp. 671-680.

Lara, C., Flores, J. and Calderón, F. 2008. Solving a School Timetabling Problem Using a Bee Algorithm. In: *7th Mexican International Conference on Artificial Intelligence: Advances in Artificial Intelligence*. Mexico: Springer. pp. 664-674.

Lawrence, D. 1985. Job Shop Scheduling with Genetic Algorithms. In: *1st International Conference on Genetic Algorithms*. L. Erlbaum Associates Inc. pp. 136-140.

Lee, C. C. 1990a. Fuzzy logic in control systems: fuzzy logic controller. I. *IEEE Transactions on Systems, Man and Cybernetics* 20(2), pp. 404-418.

Lee, C. C. 1990b. Fuzzy logic in control systems: fuzzy logic controller. II. *IEEE Transactions on Systems, Man and Cybernetics* 20(2), pp. 419-435.

Lee, J. Y. and Haj Darwish, A. 2008. Multi-objective Environmental/Economic Dispatch Using the Bees Algorithm with Weighted Sum. In: *EU-Korea Conference on Science and Technology (EKC2008)*. Heidelberg, Germany: Springer. pp. 267-274.

Lee, M. A. and Smith, M. H. 1994. Automatic Design and Tuning of a Fuzzy System for Controlling the Acrobot using Genetic Algorithms, DSFS, and Meta-Rule Techniques. In: *1st International Joint Conference of the North American Fuzzy Information Processing Society Biannual Conference, the Industrial Fuzzy Control and Intelligent Systems Conference and the NASA Joint Technology* San Antonio, TX, USA: IEEE. pp. 416-420.

Lei, B. 1999. *HYBIRD INTELLIGENT SYSTEM FOR CONDITION MONITORING AND DIAGNOSIS SYSTEM OF A CENTRIFUGAL PUMP SYSTEM.* PhD thesis, The Norwegian University of Science and Technology (NTNU).

Leu, M. C., Wong, H. and Ji, Z. 1993. Planning of Component Placement/insertion Sequence and Feeder Setup in PCB Assembly Using Genetic Algorithm. *ASME Trans, Journal of Electronic Packaging* 115(4), pp. 424-432.

Maimon, O. Z. and Brha, D. 1998. A Genetic Algorithm Approach to Scheduling PCBs on a Single Machine. *International Journal of Production Research* 36(3), pp. 761- 784.

Mamdani, E. H. and Assilian, S. 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies* 7(1), pp. 1-13.

Martinsen, D. and Krey, V. 2008. Compromises in energy policy-Using fuzzy optimization in an energy systems model. *Energy Policy* 36(8), pp. 2983-2994.

Mathur, M., Karale, S. B., Priye, S., Jayaraman, V. K. and Kulkarni, B. D. 2000. Ant Colony Approach to Continuous Function Optimization. *Industrial & Engineering Chemistry Research* 39(10), pp. 3814-3822.

Matsuoka, K., Ohyama, N., Watanabe, A. and Ooshima, M. 2006. Control of a giant swing robot using a neural oscillator. *International Congress Series* 1291, pp. 153-156.

Mazzeo, S. and Loiseau, I. 2004. An Ant Colony Algorithm for the Capacitated Vehicle Routing. *Electronic Notes in Discrete Mathematics* 18, pp. 181-186.

Metaxiotis, K. S., Askounis, D. and Psarras, J. 2002. Expert systems in production planning and scheduling: A state-of-the-art survey. *Journal of Intelligent Manufacturing* 13(4), pp. 253-260.

Muddassar, F. 2008. *Bee-Inspired Protocol Engineering: From Nature to Networks*. Springer, p. 306.

Narayanaswamy, P., Bector, C. R. and Rajamani, D. 1996. Fuzzy logic concepts applied to machine-component matrix formation in cellular manufacturing. *European Journal of Operational Research* 93(1), pp. 88-97.

Negnevitsky, M. 2005. *Artificial Intelligence : A Guide to Intelligent Systems.* 2nd ed. New York: Addison-Wesley, p. 440.

Nian, Z. and Wunsch, D. C. 2003. An extended Kalman filter (EKF) approach on fuzzy system optimization problem. In: *12th IEEE International Conference on Fuzzy Systems*. St. louis, Missouri, USA: IEEE. pp. 1465-1470.

Ong, N. and Khoo, L. P. 1999. Genetic Algorithm Approach in PCB Assembly. *Integrated Manufacturing Systems* 10(5), pp. 256-265.

Packianather, M. S. and Drake, P. R. 2005. Identifying defects on plywood using a minimum distance classifier and a neural network. In: *1st Virtual International Conference on Intelligent Production Machines and Systems (IPROMS 2005).* Oxford:Elsevier. pp. 543-548.

Passino, K., Seeley, T. and Visscher, P. 2008. Swarm Cognition in Honey Bees. *Behavioral Ecology and Sociobiology* 62(3), pp. 401-414.

Passino, K. M. and Yurkovich, S. 1998. *Fuzzy Control.* Menlo Park, CA, USA: Addison-Wesley Longman, p. 475.

Pelta, D. A., Blanco, A. and Verdegay, J. L. 2000. A fuzzy adaptive neighborhood search for function optimization. In: *4th International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies.* Brighton, UK: IEEE Press. pp. 594-597.

Petrovic, S., Fayad, C., Petrovic, D., Burke, E. and Kendall, G. 2008. Fuzzy job shop scheduling with lot-sizing. *Annals of Operations Research* 159(1), pp. 275-292.

Pham, D. T., Afify, A. A. and Koç, E. 2007a. Manufacturing cell formation using the Bees Algorithm. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*. Whittles, Dunbeath, Scotland. pp. 523-528.

Pham, D. T., AL-Jabbouli, H., Mahmuddin, M., Otri, S. and Haj Darwish, A. 2008a. Application of the Bees Algorithm to Fuzzy Clustering. In: *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008)*. Whittles, Dunbeath, Scotland. pp. 404-408.

Pham, D. T. and Alcock, R. J. 1996. Automatic detection of defects on birch wood boards. *Proceedings of the Institution of Mechanical Engineers. Part E, Journal of Process Mechanical Engineering* 210(15), pp. 45-52.

Pham, D. T., Ang, M. C., Ng, K. W., Otri, S. and Haj Darwish, A. 2008b. Generating Branded Product Concepts: Comparing the Bees Algorithm and an Evolutionary Algorithm. In: *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008)*. Whittles, Dunbeath, Scotland. pp. 398-403.

Pham, D. T., Awadalla, M. and Eldukhri, E. 2007b. Adaptive and cooperative mobile robots. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 221(3), pp. 279-293.

Pham, D. T., Castellani, M. and Fahmy, A. A. 2008c. Learning the Inverse Kinematics of a Robot Manipulator using the Bees Algorithm. In: *17th IFAC World Congress*. COEX, South Korea. pp. 493-498.

Pham, D. T., Castellani, M. and Ghanbarzadeh, A. 2007c. Preliminary Design Using the Bees Algorithm. In: *8th International Conference on Laser Metrology, CMM and Machine Tool Performance, LAMDAMAP*. Cardiff, UK: Euspen. pp. 420-429.

Pham, D. T., Castellani, M., Sholedol, M. and Ghanbarzadeh, A. 2008d. The Bees Algorithm and Mechanical Design Optimisation. In: *5th International Conference on Informatics in Control, Automation and Robotics ICINCO*. Funchal, Portugal.

Pham, D. T., Fahmy, A. A. and Eldukhri, E. E. 2008e. Adaptive Fuzzy Neural Network for Inverse Modeling of Robot Manipulators. In: *17th IFAC World Congress*. COEX, South Korea. pp. 5308-5313.

Pham, D. T. and Ghanbarzadeh, A. 2007. Multi-Objective Optimisation using the Bees Algorithm. In: *3rd International Virtual Conference on Intelligent*

Production Machines and Systems (IPROMS 2007). Whittles, Dunbeath, Scotland. pp. 529-533.

Pham, D. T., Ghanbarzadeh, A., Koc, E. and Otri, S. 2006a. Application of the Bees Algorithm to the Training of Radial Basis Function Networks for Control Chart Pattern Recognition. In: *5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering (CIRP ICME '06)*. Ischia, Italy. pp. 711-716.

Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S. and Zaidi, M. 2005. *The Bees Algorithm-Technical Report*. Cardiff: Manufacturing Engineering Centre, Cardiff University.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S. and Zaidi, M. 2006b. The Bees Algorithm - A Novel Tool for Complex Optimisation Problems. In: Pham, D.T. et al. eds. *2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS 2006)*. Elsevier, Oxford. pp. 454-459.

Pham, D. T., Ghanbarzadeh, A., Otri, S. and Koç, E. 2009. Optimal design of mechanical components using the Bees Algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 223(C5), pp. 1051-1056.

Pham, D. T. and Karaboga, D. 1999. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks.* London, Heidelberg and New York: Springer Verlag, p. 302.

Pham, D. T., Koc, E., Ghanbarzadeh, A. and Otri, S. 2006c. Optimisation of the weights of multi-layered perceptrons using the Bees Algorithm. In: *5th International Symposium on Intelligent Manufacturing Systems*. Sakarya, Turkey. pp. 38-46.

Pham, D. T., Koç, E., Kalyoncu, M. and Tınkır, M. 2008f. A Hierarchical PID Controller Design for a Flexible Link Robot Manipulator Using the Bees Algorithm. In: *6th INTERNATIONAL SYMPOSIUM on INTELLIGENT and MANUFACTURING SYSTEMS*. Sakarya, Turkey. pp. 757-765.

Pham, D. T., Koc, E., Lee, J. Y. and Phrueksanant, J. 2007d. Using the Bees Algorithm to schedule jobs for a machine. In: *8th international Conference on Laser Metrology, CMM and Machine Tool Performance (LAMDAMAP)*. Cardiff: Euspen, UK. pp. 430-439.

Pham, D. T., Lee, J. Y., Haj Darwish, A. and Soroka, A. J. 2008g. Multi-objective Environmental/Economic Power Dispatch using the Bees Algorithm with Pareto optimality. In: *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008)*. Whittles, Dunbeath, Scotland. pp. 422-430.

Pham, D. T. and Liu, X. 1995. *Neural Networks for Identification, Prediction and Control.* New York: Springer, p. 238.

Pham, D. T., Mahmuddin, M., Otri, S. and Al-Jabbouli, H. 2007e. Application of the Bees Algorithm to the Selection Features for Manufacturing Data. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007).* Whittles, Dunbeath, Scotland. pp. 517-522.

Pham, D. T., Muhamad, Z., Mahmuddin, M., Ghanbarzadeh, A., Koc, E. and Otri, S. 2007f. Using the Bees Algorithm to Optimise a Support Vector Machine for Wood Defect Classification. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007).* Whittles, Dunbeath, Scotland. pp. 540-545.

Pham, D. T., Otri, S., Afify, A., Mahmuddin, M. and Al-Jabbouli, H. 2007g. Data Clustering Using the Bees Algorithm. In: *40th CIRP International Manufacturing Systems Seminar.* Liverpool, UK.

Pham, D. T., Otri, S., Ghanbarzadeh, A. and Koc, E. 2006d. Application of the Bees Algorithm to the Training of Learning Vector Quantisation Networks for Control Chart Pattern Recognition. In: *2nd IEEE International Conference on Information and Communication Technologies:From Theory to Applications.* Damascus, Syria. pp. 1624-1629.

Pham, D. T., Otri, S. and Haj Darwish, A. 2007h. Application of the Bees Algorithm to PCB assembly optimisation. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*. Whittles, Dunbeath, Scotland. pp. 511-516.

Pham, D. T., Pham, Q. T., Ghanbarzadeh, A. and Castellani, M. 2008h. Dynamic Optimisation of Chemical Engineering Processes Using the Bees Algorithm. In: *17th IFAC World Congress*. COEX, South Korea. pp. 6100-6105.

Pham, D. T. and Sholedolu, M. 2006. Multi-layer perceptron network training for Control Chart Pattern Recognition using the particle swarm optimisation algorithm. In: *5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering (CIRP ICME '06)*. Ischia, Italy. pp. 717-722.

Pham, D. T. and Sholedolu, M. 2008. Using a Hybrid PSO-Bees Algorithm to train Neural Networks for Wood Defect Classification. In: *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008)*. Whittles, Dunbeath, Scotland. pp. 385-390.

Pham, D. T., Soroka, A., Koc, E., Ghanbarzadeh, A. and Otri, S. 2007i. Some Applications of the Bees Algorithm in Engineering Design and Manufacture. In: *International Conference on Manufacturing Automation (ICMA '07)*. Singapore. pp. 782-794.

Pham, D. T., Soroka, A. J., Ghanbarzadeh, A., Koc, E., Otri, S. and Packianather, M. 2006e. Optimising Neural Networks for Identification of Wood Defects Using the Bees Algorithm. In: *IEEE International Conference on Industrial Informatics*. Singapore. pp. 1346-1351.

Quijano, N. and Passino, K. M. 2007a. Honey Bee Social Foraging Algorithms for Resource Allocation, Part I: Algorithm and Theory. In: *American Control Conference, ACC '07*. New York, USA. pp. 3383-3388.

Quijano, N. and Passino, K. M. 2007b. Honey Bee Social Foraging Algorithms for Resource Allocation, Part II: Application. In: *American Control Conference, ACC '07*. New York, USA. pp. 3389-3394.

Reardon, B. J. 1998. Fuzzy logic versus niched Pareto multiobjective genetic algorithm optimization. *Modelling and Simulation in Materials Science and Engineering* 6(6), pp. 717-734.

Rooij, A. J. F. V., Johnson, R. P. and Jain, L. C. 1996. *Neural Network Training Using Genetic Algorithms*. River Edge, NJ, USA: World Scientific Publishing Co, p. 130.

Russell, S. and Norvig, P. 2004. *Artificial Intelligence: A Modern Approach*. 2nd Edition ed. Upper Saddle River, NJ, USA: Prentice-Hall, p. 1132.

Saad, E. M., Awadalla, M. H. and Darwish, R. R. 2008a. Adaptive Energy-Aware Gathering Strategy for Wireless Sensor Networks. *INTERNATIONAL JOURNAL OF COMPUTERS* 2(2), pp. 148-157.

Saad, E. M., Awadalla, M. H. and Darwish, R. R. 2008b. A Data Gathering Algorithm for a Mobile Sink in Large-Scale Sensor Networks. In: *IEEE 4th International Conference onWireless and Mobile Communications (ICWMC '08).* Athens, Greece: IEEE Computer Society. pp. 207-213.

Saad, S. and Khalil, E. 2005. Taboo Search vs Genetic Algorithms in Solving and Optimising PCB Problems. *Journal of Manufacturing Technology Management* 17(4).

Sahran, S. 2007. *APPLICATION OF SPIKING NEURAL NETWORKS AND THE BEES ALGORITHM TO CONTROL CHART PATTERN RECOGNITION.* PhD thesis, Cardiff University.

Sanii, E. T. and Liau, J.-S. 1993. An expert process planning system for electronics PCB assembly. *Computers and Electrical Engineering* 19(2), pp. 113-127.

Seeley, T. D. 1996. *The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies*. Cambridge, Massachusetts: Harvard University Press, p. 318.

Seeley, T. D., Visscher, P. K. and Passino, K. M. 2006. Group Decision Making in Honey Bee Swarms. *American Scientist* 94(3), pp. 220-229.

Seo, M. and Kim, D. 2009. Ant colony optimisation with parameterised search space for the job shop scheduling problem. *International Journal of Production Research* (1), pp. 1-12.

Shang, G., Lei, Z., Fengting, Z. and Chunxian, Z. 2007. Solving Traveling Salesman Problem by Ant Colony Optimization Algorithm with Association Rule. In: *3rd International Conference on Natural Computation (ICNC 2007).* Haikou, Hainan, China: IEEE Computer Society. pp. 693-698.

Shankir, Y. M. A. A. 2000. *Fuzzy logic systems and fuzzy neural networks for dynamic systems modelling and control.* PhD thesis, Cardiff University.

Shaout, A. and McAuliffe, P. 1998. Job scheduling using fuzzy load balancing in distributed system. *Electronics Letters* 34(20), pp. 1983-1985.

Shaout, A. and McAuliffe, P. 2000. Fuzzy load balancing in an industrial distributed system for job scheduling. *Journal of Intelligent and Fuzzy Systems* 8(3), pp. 191-199.

Sheibani, K. 2006. Fuzzy Greedy Search and Job-Shop Problem. In: *25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG).* Nottingham, UK pp. 147-150.

Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C. and Wang, Q. X. 2007. Particle swarm optimization-based algorithms for TSP and generalized TSP. *Information Processing Letters* 103(5), pp. 169-176.

Shi, Y. and Eberhart, R. 1998a. A modified particle swarm optimizer. In: *IEEE International Conference on Evolutionary Computation*. Anchorage, Alaska: IEEE. pp. 69-73.

Shi, Y. and Eberhart, R. 1998b. Parameter Selection in Particle Swarm Optimization. In: *7th International Conference on Evolutionary Programming VII*. San Diego, California, USA: Springer. pp. 591-600.

Shih, W., Srihari, K. and Adriance, J. 1996. Expert system based placement sequence identification for surface mount PCB assembly. *The International Journal of Advanced Manufacturing Technology* 11(6), pp. 413-424.

Silva, L., Torres, G., Reis, L. and Haddad, J. 2002. Application of fuzzy optimization in energy saving. *Revista Ciencias Exatas* 8(2), pp. 21- 35.

Simon, D. 2002a. Training fuzzy systems with the extended Kalman filter. *Fuzzy Sets and Systems* 132(2), pp. 189-199.

Simon, D. 2002b. Training radial basis neural networks with the extended Kalman filter. *Neurocomputing* 48(1-4), pp. 455-475.

Smith, M. H., Zhang, T. and Gruver, W. A. 1998. Dynamic fuzzy control and system stability for the Acrobot. In: *IEEE International Conference on Fuzzy Systems*. Anchorage, AK, USA: IEEE. pp. 286-291.

Spall, J. C. 2003. *Introduction to Stochastic Search and Optimization*. New York: Wiley, p. 618.

Spong, M. W. 1995. The Swing up Control Problem for the Acrobot *IEEE Control System Magazine* 15(2), pp. 49-55.

Stanciulescu, C., Fortemps, P., Installé, M. and Wertz, V. 2003. Multiobjective fuzzy linear programming problems with fuzzy decision variables. *European Journal of Operational Research* 149(3), pp. 654-675.

Steeb, W.-H. 2008. *The nonlinear workbook : chaos, fractals, cellular automata, neural networks, genetic algorithms, gene expression programming, support vector machine, wavelets, hidden Markov models, fuzzy logic with C++, Java and SymbolicC++ programs*. 4th ed. Singapore: World Scientific, p. 605.

Su, Y.-Y. and Srihari, K. 1996. Placement sequence identification using artificial neural networks in surface mount PCB assembly. *The International Journal of Advanced Manufacturing Technology* 11(4), pp. 285-299.

Sugeno, M. 1985. *Industrial Applications of Fuzzy Control.* New York: Elsevier, p. 269.

Takahashi, R. 2005. Solving the traveling salesman problem through genetic algorithms with changing crossover operators. In: *4th International Conference on Machine Learning and Applications.* Los Angeles, California, USA: IEEE Computer Society. pp. 319-324.

Tanaka, k. 1997. *An introduction to fuzzy logic for practical applications.* New York: Springer, p. 138.

Teodorovic, D. and Dell'orco, M. 2005. Bee colony optimization - A cooperative learning approach to complex transportation problems. *Advanced OR and AI methods in transportation*, pp. 51-60.

Timothy, M. 1993. *Practical neural network recipes in C++.* San Diego, CA, USA: Academic Press Professional, Inc, p. 493.

Tovey, C. A. 2004. *The Honey Bee Algorithm: A Biologically Inspired Approach to Internet Server Optimization.* Engineering Enterprise Magazine. pp. 13-15.

Tsoukalas, L. H. and Uhrig, R. E. 1997. *Fuzzy and Neural Approaches in Engineering.* NewYork: Wiley, p. 587.

Tu, K., Hao, Z. and Chen, M. 2006. PSO with Improved Strategy and Topology for Job Shop Scheduling. In: *2nd International Conference on Natural Computation*. Xi'an, China: Springer. pp. 146-155.

Valluru, B. R. and Hayagriva, R. 1995. *C++ neural networks and fuzzy logic.* 2nd ed. New York, USA: MIS-Press, p. 551.

Wang, J., Zhao, H., Du, J., Yu, T. and Wang, W. 2008. Neural Network Model Based Job Scheduling and Its Implementation in Networked Manufacturing. In: *4th IEEE International Conference on Natural Computation, ICNC '08*. Jinan, Shandong, China: IEEE Computer Society. pp. 480-484.

Wang, J., Zhu, L., Cai, Z., Gong, W. and Lu, X. 2007. Training RBF Networks with an Extended Kalman Filter Optimized Using Fuzzy Logic. In: *IFIP TC12 International Conference on Intelligent Information Processing* Adelaide, Australia: Springer. pp. 317-326.

Wang, T. J. and Chaharbaghi, K. 1995. PLANNING CELLULAR MANUFACTURING SYSTEMS USING FUZZY LOGIC. In: *IEE Colloquium on Intelligent Manufacturing Systems*. IEE Institution of Electrical Engineers. p. 6.

Wedde, H. F., Farooq, M. and Zhang, Y. 2004. BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior. In: O.Jones, K. ed.

4th International Workshop on Ant Colony, Optimization and Swarm Intelligence. Brussels, Belgium: Springer. pp. 83-94.

Wei, P., Kang-ping, W., Chun-guang, Z. and Dong, L.-j. 2004. Fuzzy discrete particle swarm optimization for solving traveling salesman problem. In: Kang-ping, W. ed. *4th IEEE International Conference on Computer and Information Technology, CIT '04*. Wuhan, China: IEEE Computer Society. pp. 796-800.

Weiss, S. M. and Kulikowski, C. A. 1991. *Computer systems that learn : classification and prediction methods from statistics, neural nets, machine learning and expert systems*. San Mateo, California: Kaufmann Publishers, p. 223.

Wiklendt, L., Chalup, S. and Middleton, R. 2008. A small spiking neural network with LQR control applied to the acrobot. *Neural Computing & Applications* 18(4), pp. 369-375.

Wong, H. and Leu, M. C. 1993. Adaptive Genetic Algorithm for Optimal Printed Circuit Board Assembly Planning. *Annals of the CIRP* 42, pp. 17-20.

Yang, X.-S. 2005. Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. In: *1st International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC 2005)*. Las Palmas, Canary Islands, Spain: Springer. pp. 317-323.

Yang, X.-S. 2008. *Nature-Inspired Metaheuristic Algorithms*. Frome, UK: Luniver Press, p. 128.

Ye, J., Qiao, J., Li, M.-a. and Ruan, X. 2007. A tabu based neural network learning algorithm. *Neurocomputing* 70(4-6), pp. 875-882.

Yen, J. and Langari, R. 1999. *Fuzzy logic: intelligence, control, and information*. New Jersey: Prentice-Hall, p. 548.

Yu, L., Liu, K. and Li, K. 2007. Ant colony optimization in continuous problem. *Frontiers of Mechanical Engineering in China* 2(4), pp. 459-462.

Zadeh, L. A. 1965. Fuzzy sets. *Information and Control* 8(3), pp. 338-353.

Zhao, D.-b. and Yi, J.-q. 2006. A Particle Swarm Optimized Fuzzy Neural Network Control for Acrobot. In: *3rd International Symposium on Neural Networks*. Chengdu, China: Springer. pp. 1160-1165.

Zhou, T., Jia, Z. and Liu, X. 2007. A Novel Chaotic Neural Network for Function Optimization. In: *14th International Conference on Neural Information Processing, ICONIP*. Kitakyushu, Japan: Springer. pp. 426-433.

Zimmermann, H. J. 1996. *Fuzzy set theory: and its applications*. 3rd edition ed. Boston: Kluwer Academic Publishers, p. 560.