HYBRID INTELLIGENT OPTIMIZATION TECHNIQUES AND ITS INDUSTRIAL APPLICATIONS

LAI CHUNG YEE JOHNNY

Ph.D

The Hong Kong
Polytechnic University

2014

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

HYBRID INTELLIGENT OPTIMIZATION TECHNIQUES AND ITS INDUSTRIAL APPLICATIONS

by

LAI CHUNG YEE JOHNNY

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

August 2013

STATEMENT OF ORIGINALITY

The following contributions reported in this thesis are claimed to be original.

1. Applying a wavelet function to determinate the weighting value *F* in Differential Evolution mutation (Chapter 3). Owing to the properties of the wavelet, the searching performance of Differential Evolution is improved.
2. Embedding a wavelet based mutation operation in Differential Evolution to modify the trial vector in the searching process (Chapter 3). Thanks to the wavelet properties, the searching region can be controlled and searching performance can be improved.
3. Combining 1) and 2) to formulate an improved Differential Evolution algorithm called the Differential Evolution with Double Wavelet Mutations (DWM-DE) (Chapter 3). The proposed DWM-DE can provide better performance in term of solution quality, solution reliability, and convergence rate.
4. Applying the proposed DWM-DE algorithm on the Economic Load Dispatch (ELD) problem (Chapter 5). The DWM-DE algorithm offers good performance on obtaining a low operating cost in the ELD problem.
5. Applying the proposed DWM-DE algorithm to train a fuzzy inference system for detecting hypoglycaemia (Chapter 6). The DWM-DE algorithm offers good results on the training

process for constructing the detection model.

6. Developing an intelligent optimizer that embeds two Differential Evolution engines into one single system (Chapter 4). By using T-test to analyse the population difference between two systems and a fuzzy controller, the optimiser adjusts the internal parameters of the two engines adaptively. The proposed intelligent optimiser can provide better performance in term of solution quality, solution reliability, and convergence rate.
7. Applying the proposed intelligent optimiser on the Economic Load Dispatch (ELD) problem (Chapter 5). The intelligent optimiser offers good performance on obtaining a low operating cost in the ELD problem.
8. Applying the proposed intelligent optimiser to train a fuzzy inference system for detecting hypoglycaemia (Chapter 6). The intelligent optimiser offers good results on the training process for constructing the detection model.

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in text.

_____ (Signed)

LAI CHUNG YEE JOHNNY (Name of Student)

ABSTRACT

This thesis focuses on developing efficient methods to solve different real-world optimisation problems. The proposed methods are based on Evolutionary Computation (EC) to perform the optimisation. Results in the following areas will be reported. (1) An improved Differential Evolution with Double Wavelet Mutations (DWM-DE) is proposed as a general-purpose evolutionary algorithm. (2) An improved intelligent optimiser that integrates two optimisation engines is proposed to solve high-dimension and complex optimisation problems. (3) The industrial optimisation problem of Economic Load Dispatch with Valve-Point Loading (ELD-VPL) is used to test the performance of the proposed methods in (1) and (2). (4) The biomedical application of hypoglycaemia detection is employed as a real-world complex classification platform for testing the performance of the proposed methods in (1) and (2).

In this thesis, an improved optimisation algorithm and an intelligent optimiser are proposed for high-dimension complex optimisation problems. The algorithm is an improved version of Differential Evolution (DE) called DE with double wavelet mutations (DWM-DE). By introducing the double wavelet mutations in DE, the searching process is enhanced by offering an effective balance between the exploration and exploitation of the solution space for better solution reliability and quality. In the DE mutation operation, a wavelet function is employed to control the mutation factor F. In the DE crossover operation, a wavelet-based second mutation mechanism is proposed to modify the trial vectors within the population. A suite of 29 benchmark test functions is employed to test the performance of the proposed DWM-DE. The experiment results show that the proposed DWM-DE is a useful tool for solving

optimisation problems, and it offers better results in terms of solution reliability, solution quality and convergence rate. The experiment results reflect that DWM-DE is particularly suitable for complex problems with a high dimension.

The intelligent optimiser embeds two DE engines into one single system. Through sharing the population information of the two DE engines, the optimiser offers better searching performance. The user of the intelligent optimiser is not required to set the parameter values of the optimiser. The two DE engines operate in parallel and an internal fuzzy controller is employed to adjust the parameter values adaptively in real time during the iteration process. The fuzzy controller takes the searching process information of the population as input. The Student T-Test method is employed to obtain the difference of the population information in the two engines. The resulting intelligent optimiser is capable of dealing with different high-dimension complex optimisation problems efficiently. A suite of 29 benchmark test functions is employed to test the performance of the proposed intelligent optimiser. The experiment results show that the proposed intelligent optimiser a useful tool for solving optimisation problems, and it offers better results in terms of solution reliability, solution quality and convergence rate. In particular, the experiment results show that the intelligent optimiser could offer much better results when the problem is complex and the problem dimension is high (>30).

The ELD-VPL problem concerns the process of sharing the power demand among online generators in a power system for the minimum fuel cost. The proposed DWM-DE algorithm and intelligent optimiser are employed to solve the ELD-VPL problem. Two different ELD-VPL problems of different scales have been tested. It is observed that the proposed methods give better optimal costs when compared with other techniques in the literature.

A fuzzy inference system (FIS) is employed as a classifier to classify the presence of hypoglycaemic episodes for Type 1 diabetes mellitus (TIDM) patients by measuring some physiological signals continuously from human body. It captures the relationship between the presence of hypoglycaemic episodes and the physiological signals of corrected QT interval of the electrocardiogram (ECG) signal and heart rate. The proposed DWM-DE algorithm and intelligent optimiser are employed to optimise the FIS parameter values that formulate the fuzzy rules and fuzzy membership functions. Data of 15 children with TIDM are studied and used in the training and testing process for the proposed FIS. The experiment results show that the two proposed optimisation methods could offer good performance on training the FIS. The resulting FIS can offer good performance on doing hypoglycaemia detection.

AUTHOR'S PUBLICATIONS

International Journal Papers

1) **J.C.Y. Lai**, F.H.F. Leung, S.H. Ling and H.T. Nguyen, "**Hypoglycaemia detection using fuzzy inference system with multi-objective double wavelet mutation differential evolution**," *Applied Soft Computing*, vol. 13, issue 5, pp. 2803-2811, May 2013.
2) **J.C.Y. Lai**, F.H.F. Leung, S.H. Ling and E. Shi, "**An improved differential evolution and its industrial application**," *Journal of Intelligent Learning Systems and Applications*, vol. 4, no. 2, pp. 81-97, May 2012.
3) **J.C.Y. Lai**, F.H.F. Leung, S.H. Ling and H.T. Nguyen, "**Hypoglycaemia detection using fuzzy inference system with intelligent optimizer**," *Applied Soft Computing* (under revision).

International Conference Papers

1) **J.C.Y. Lai**, F.H.F. Leung, S.H. Ling and E.C. Shi "**Economic load dispatch using intelligent optimization with fuzzy control**," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE 2011)*, Taipei, Jun. 27-30, 2011, pp. 2219-2224.
2) E.C. Shi, F.H.F. Leung, and **J.C.Y. Lai**, "**An adaptive differential evolution with unsymmetrical mutation**," in *Proc. 2011 IEEE Congress on Evolutionary Computation (CEC 2011)*, New Orleans, USA, Jun. 5-8, 2011.
3) **J.C. Lai**, F.H. Leung, and S.H. Ling, "**A new differential evolution with self-terminating ability using fuzzy control and k-nearest neighbors**," in *Proc. 2010 IEEE Congress on*

Evolutionary Computation (CEC 2010), World Congress on Computational Intelligence (WCCI 2010), Barcelona, Spain, Jul. 18-23, 2010, pp. 503-510.

4) **J.C. Lai**, F.H. Leung, and S.H. Ling, "**Economic load dispatch using differential evolution with double wavelet mutation operations**," in *Proc. 2010 IEEE Congress on Evolutionary Computation (CEC 2010), World Congress on Computational Intelligence (WCCI 2010)*, Barcelona, Spain, Jul. 18-23, 2010, pp. 2013-2018.

5) **J.C.Y. Lai**, C.W. Yeung, and F.H.F. Leung, "**A new hybrid differential evolution with wavelet based mutation and crossover**," in *Proc. Asia-Pacific Signal and Information Processing Association 2009 Annual Summit and Conference (APSIPA ASC 2009)*, Sapporo, Japan, Oct. 4-7, 2009, pp. 722-729.

6) C.W. Yeung, **J.C.Y. Lai**, and F.H.F. Leung, "**A particle swarm optimization with multi-mutation operation base on wavelet theory**," in *Proc. Asia-Pacific Signal and Information Processing Association 2009 Annual Summit and Conference (APSIPA ASC 2009)*, Sapporo, Japan, Oct. 4-7, 2009, pp. 730-737.

7) **J.C.Y. Lai**, F.H.F. Leung and S.H. Ling, "**A new differential evolution with wavelet theory based mutation operation**," in *Proc. 2009 IEEE Congress on Evolutionary Computation (CEC 2009)*, Trondheim, Norway, May 18-21, 2009, pp. 1116-1122.

ACKNOWLEDGEMENT

I would like to express my deep gratitude to my project chief supervisor, Dr Frank H.F. Leung, for his continuous support on my research study. Without his valuable advice, my research study cannot be done successfully. In addition, I would also like to thank my Co-supervisor, Dr Steve S.H. Ling, for his helpful advice and suggestions on my work.

I would like to express my special thanks to my family and Ms Cathy T.T. Sze for their support and continual encouragement throughout my study. My gratitude is extended to my friends, Miss Ginny Wong, Mr Benny C.Y. Yeung and Mr Ivan Lau, for their help in my research work. I would also like to show my great appreciation to the staff in the General Office of the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, for their administrative supports.

The work described in this thesis was substantially supported by a grant from the Hong Kong Polytechnic University of the Hong Kong Special Administrative Region (PhD Project Account Code RP9L).

TABLE OF CONTENTS

LIST OF SYMBOLS

ΔHR	Change of heart rate
ΔQT_c	Change of corrected QT interval of the electrocardiogram signal
a	Dilation parameter of multi-wavelet function
C_r	Crossover rate of Differential Evolution
F	Weighting factor of Differential Evolution
f_m	Fuzzy membership function
h	Binary state of hypoglycaemia
HR	Heart rate
N_{FN}	Number of false negative
N_{FP}	Number of false positive
N_{TN}	Number of true negative
N_{TP}	Number of true positive
P	Population in Differential Evolution
QT_c	Corrected QT interval of the electrocardiogram signal
T	Total number of iterations
t	Current number of iteration
u	Crossover vector in Differential Evolution
v	Mutated vector in Differential Evolution
α	Mean
ζ_{wm}	Shape parameter of wavelet mutation

η_{spec}	Specificity
λ	Upper limit of the dilation parameter of a multi-wavelet function
ξ_{degree}	Degree of freedom in T-Test
ξ_{sen}	Sensitivity
σ	Standard deviation
$\psi(\cdot)$	Mother wavelet function
$\psi_{a,b}(\cdot)$	Multi-wavelet function

ABBREVIATIONS

CI	**C**omputational **I**ntelligence
CPU	**C**entral **P**rocessing **U**nit
DE	**D**ifferential **E**volution
DE/BBO	Hybrid **D**ifferential **E**volution With **B**iogeography-**B**ased **O**ptimization
DEC-SQP	**D**ifferential **E**volution **C**ombination with **S**equential **Q**uadratic **P**rogramming Method
DWM-DE	**D**ifferential **E**volution with **D**ouble **W**avelet **M**utations
EC	**E**volutionary **C**omputation
ECG	**E**lectro**c**ardio**g**ram
ELD	**E**conomic **L**oad **D**ispatch
EP	**E**volutionary **P**rogramming
FFNN	**F**eed-**F**orward **N**eural **N**etwork
FIS	**F**uzzy **I**nference **S**ystem
GA	**G**enetic **A**lgorithm
HGAPSO	**H**ybrid **G**enetic **A**lgorithm **P**article **S**warm **O**ptimization
HGPSO	**H**ybrid **G**radient Descent **P**article **S**warm **O**ptimization
HPSOM	**H**ybrid **P**article **S**warm **O**ptimization with **M**utation
HR	**H**eart **R**ate
IGA	Adaptive-**I**mproved **G**enetic **A**lgorithm
MPSO	**M**odified **P**article **S**warm **O**ptimization

NN	**N**eural **N**etwork
NPSO-LRS	**N**ew **P**article **S**warm **O**ptimization with **L**ocal **R**andom **S**earch
PSO	**P**article **S**warm **O**ptimization
QPSO	**Q**uantum **P**article **S**warm **O**ptimization
RCGA	**R**eal-**C**oded **G**enetic **A**lgorithm
SOH-PSO	**S**elf-**O**rganizing **H**ierarchical **P**article **S**warm **O**ptimization
SWM-DE	**D**ifferential **E**volution with **S**ingle **W**avelet **M**utation
T1DM	**T**ype **1** **D**iabetes **M**ellitus
VPL	**V**alve **P**oint **L**oading
WM-DE	**W**avelet **M**utated **D**ifferential **E**volution

Chapter 1

INTRODUCTION

I BACKGROUND

Many real-world applications in different fields face a common problem of optimisation. The major goal of doing optimisation is to find the best solution of a given problem within a defined domain. We could describe an optimisation problem by specifying the possible set of feasible candidate solutions and measure the possible set based on different problem conditions. Mathematically, the choice of the best solution usually depends on a defined objective function, possibly subject to some constraints. In the simplest case, we seek to minimise or maximise a real-valued objective function by systematically choosing the input values of the objective function from an allowed set in a given domain. In the fields of science and engineering, optimisation is the process to minimize a system's undesirable properties and maximize its desirable properties. What the details of the system properties are and how the system performance can be improved often depend on the information given by and the expert knowledge on the system.

In the past decade, many different kinds of algorithms have been developed to perform optimisation. However, most of them have their own limitations when applied to different

problems. As a result, there is growing interest on improving algorithms to solve optimisation problems. Many researchers have been working for methods to explore and exploit the hidden problem features for solving various real-world problems in an efficient and scalable way [Chiong 12] [Bosman 03] [Pelikan 02]. It is a kind of black-box approach, which means optimising a system without the presence of an accurate algebraic model for that system. Yet, some expert knowledge of the problem can be obtained by capturing the relationship between the candidate solutions and their suitability to the problem. There are many well-known methods for realising black-box optimisation; for example, the gradient descent search, golden section search, Fibonacci search, hill climbing search, etc. [Avriel 03]. A well-structured traversal of the search space incorporating the state-of-the-art computing algorithms within the area of Computational Intelligence (CI) has been employed to realise optimisation. CI techniques have been actively researched in order to improve their performance on complex optimisation problems.

Thanks to the rapid growth of computer hardware and technologies, Evolutionary Computation (EC) algorithms, which are kinds of CI techniques, have been well accepted as reliable methods for tackling complex optimisation problems. This is particularly true when the objective function to be optimised is a multi-modal function with many local optima and is non-differentiable. EC adopts an iterative and progressive approach, such as growing or developing a population, to reach the optimised solution. In each iteration step, the population is updated through a guided random search that aims at the desired end. Such a process of iterative searching is often inspired by the biological mechanisms of evolution. Popular examples of EC algorithms for tackling optimisation problems include Particle Swarm Optimisation (PSO), Genetic Algorithm (GA), Ant Colony Optimisation (ACO) and Differential Evolution (DE) [Chiong 12].

Different optimisation problems have different characteristics and may require different EC algorithms for searching the best solution. Hence, determining the most suitable EC algorithm for a given optimisation problem becomes an important consideration of researchers. Moreover, many EC algorithms often require the proper setting of parameters values in order to enhance the convergence and accuracy of the solution. Example parameters to be considered include the scaling factor in Differential Evolution (DE), the acceleration constants in Particle Swarm Optimisation (PSO), the probability of mutation in Genetic Algorithm (GA), etc. Besides, on implementing an EC algorithm, the number of iteration often needs to be pre-determined by the user based on a trial-and-error approach or on expert knowledge to the problem. A wrong estimate of the number of iteration may lead to pre-mature evolution, convergence to local optimum only, or unnecessary wasting of computational power.

In recent years, a number of high-dimension complex optimisation problems emerge. Many EC methods fail to handle them. Moreover, many EC methods require a huge amount of time to obtain the best solution. Two examples of high-dimension complex optimisation problems are covered in this thesis. They are the economic load dispatch with valve-point loading problem and the hypoglycaemia detection problem. The economic load dispatch with valve-point loading problem is an industrial optimisation problem which minimizes the fuel cost of power generation by sharing the loading demand among the online power generators in a power supply system. It is a problem of as high as 40 dimensions with a complex objective function. Hypoglycaemia detection is an important problem in the area of insulin therapy. Insulin therapy is a medical treatment for patients with diabetes. A classification model is required to support the detection. To construct the classification model, an optimiser is required to reduce the detection error of the model.

In this thesis, an improved optimisation algorithm and an intelligent optimiser are proposed for high-dimension complex optimisation problems. The algorithm is an improved version of Differential Evolution (DE) called DE with double wavelet mutations (DWM-DE). The standard DE algorithm can offer good performance in many applications; but when the dimension of the application is high, the performance of DE may drop significantly. The proposed DWM-DE alleviates this problem by embedding wavelet mutations in both the DE mutation and crossover operations. In a typical optimisation problem, we have to achieve a balance between the exploration and the exploitation of the searching space. In the early stage of searching, we want more exploration while more exploitation is desired at the later stage of searching. Exploration means widely searching the solution space in a large area. Exploitation means searching the solution space in a small area to obtain a fine-tuned solution. This mechanism can be realized by a wavelet function. A wavelet function is a mathematical tool to model seismic signals in a finite domain. The balance between the exploration and exploitation can be achieved by taking advantage of the wavelet function's properties. The wavelet function is embedded in the double wavelet mutation operations that provide a more efficient searching process with better reliability in searching the global solutions of different problems.

The proposed intelligent optimiser embeds two DE engines into one single system. Through sharing the population information of the two DE engines, the optimiser offers better searching performance. The user of the intelligent optimiser is not required to set the parameter values of the optimiser. The two DE engines operate in parallel and an internal fuzzy controller is employed to adjust the parameter values adaptively in real time during the iteration process. The fuzzy controller takes the searching process information of the population as input. The Student T-Test method is employed to obtain the difference of the population information in the

two engines. The resulting intelligent optimiser is capable of dealing with different high-dimension complex optimisation problems efficiently.

II OBJECTIVES OF RESEARCH

In the light of the above discussion, our research aims at meeting the following five objectives, and the results are discussed in this thesis.

1. While many different kinds of optimisation methods have been proposed, they often can only work successfully in specific problem areas. For instance, some methods work well in unimodal problems and some work well in multimodal problems. Finding the suitable optimisation method for a given problem becomes a challenge. The user may be required to have a deep understanding of the problem nature, which could introduce great difficulty. We work on proposing improved methods that can relax the performance dependence on different problem natures.

2. Optimisation algorithms often require a proper setting of their parameter values done by the user. The performance of the algorithms might depend very much on these values. Expert knowledge on the problem nature and the optimisation algorithm is often required before the user can determine the best parameter values. We work on proposing methods that are not so dependent on the parameter values or can adaptively set the values during the iterative process.

3. Apart from making the optimisation method easy to implement and use, we should ensure its reliable convergence to the global optimum. Furthermore, the computation time required to perform the searching of the solution should not be excessive. Thus, we work on

proposing useful global optimisation methods that are simple to implement, easy to use and reliable.

4. We work on proposing methods that can handle the economic load dispatch with valve-point loading (ELD-VPL) problem.

5. We work on designing a classification model for the hypoglycaemia detection problem, and proposing suitable methods to tune the model for doing the detection.

III ORGANIZATION OF THESIS

The organization of this thesis is given as follows.

Chapter 2 provides a literature review. It introduces the current research in the CI area on solving optimisation problems, with emphasis particularly on the development of EC methods. Different kinds of EC methods are introduced in this chapter. Moreover, some findings and results about the ELD-VPL and the hypoglycaemia detection problems are also discussed.

Chapter 3 introduces the proposed Differential Evolution with Double Wavelet Mutations (DWM-DE) algorithm. By introducing the double wavelet mutations in the Differential Evolution algorithm, the searching process is enhanced by offering an effective balance between exploration and exploitation of the solution at different stages of evolution for better solution reliability and quality. The performance of the proposed DWM-DE algorithm is evaluated by using a suite of 29 benchmark test functions which covers four different categories of optimisation problems.

Chapter 4 presents the second method for tackling complex and high-dimension optimisation problems. An intelligent optimiser is proposed that integrates two DE engines into one single system. The two DE engines share their individual population information with each other to achieve better searching performance. The Student T-Test method is employed to analyse the two populations' information during the searching process, which is fed to an internal fuzzy controller as input to adjust the parameter values of the two DE engines adaptively. The proposed intelligent optimiser is also tested with the 29-benchmark test functions used in the previous chapter to evaluate its performance.

In Chapter 5, the ELD-VPL problem is introduced. It is an engineering process to share the power demand among the online generators in a power system for the minimum fuel cost. The two proposed optimisation methods are employed to solve the ELD-VPL problem. Two different requirements of the ELD-VPL problem will be tested. It is observed that the two proposed methods give satisfactory optimal costs when compared with other techniques in the literature.

Chapter 6 presents the application of hypoglycaemia detection that involves the optimisation process. To classify the present of hypoglycaemic episodes for Type 1 diabetes mellitus (TIDM) patients, a detector is developed to measure some physiological signals continuously from human body. A fuzzy inference system (FIS) is employed in the detector for doing modelling, which captures the relationship between the presence of hypoglycaemic episodes and the physiological signals of corrected QT interval of the electrocardiogram (ECG) signal and heart rate. The two proposed methods discussed in Chapters 3 and 4 are employed to optimise the FIS parameter values that formulate the fuzzy rules and fuzzy membership functions. The optimisation is formulated as a multi-objective problem. Moreover, a validation

mechanism is proposed in the training process to avoid over-fitting the training data to the proposed FIS. This phenomenon is called overtraining. The data of 15 children with TIDM are studied and used in the training and testing process for the proposed FIS. The experiment result shows that the two proposed optimisation methods can offer good performance on training the FIS. The FIS can offer good performance on doing hypoglycaemia detection.

A conclusion for this thesis will be given in Chapter 7. Some directions for future work are also presented in this chapter.

Chapter 2

LITERATURE REVIEW

Evolutionary Computation (EC) is one of the popular intelligent methods for handling optimisation problems. It is a sub-topic of Computational Intelligence (CI). Before EC became popular on solving optimisation problems, classical derivative-based non-linear optimisation techniques were the major tools on solving optimisation problems. The Pontyagrin's principle, Lagrangian relaxation, Bellman's principle, and Lagrange's Multiplier are the common tools for obtaining the best solution of a system [Swagatam 08]. Unfortunately, when the complexity and dimension of the optimisation problems increase, those derivative-based optimisation techniques fail to solve the problem with the best solution quality. The complexities could be due to rough function surfaces, discontinuous behaviour, multimodal function properties, etc. EC methods were proposed to overcome the limitations of classical non-linear optimisation techniques. Fig. 2.1 shows a summary of different optimisation method families.

EC controls a group of elements (population) to search for the optimal solution. The control of population is an iterative process to guide the population to reach the optimal point in the solution space. The population members are selected and the evolution is controlled by a designed rule in a way of guided random search [Zhang 05]. One of the successful EC methods is called Genetic Algorithm (GA), which is commonly used on solving complex optimisation problems [Ahn 02].

The searching operation of GA is inspired by the natural adaptation of biological species [Michalewicz 94]. It simulates the biological crossover and mutation operations of chromosomes in the searching operation and enhances the quality of the solution by a number of evolutions [Srinivas 94]. GA is implemented such that a population of chromosomes (candidate solutions) on the solution domain evolve toward better solutions [Zhang 07]. The evolution happens in a number of generations (iteration steps). GA evaluates the fitness of every chromosome in the population in each generation. This process is a measure of the suitability of a chromosome. After taking some evolutionary operations, the better (more fitted) individuals in the population are selected and a new population is formed as the next generation. The new population then repeat the evolution process [Michalewicz 94]. GA terminates the searching process with a defined number of iterations. Different problem areas works with GA successfully, including the tuning of fuzzy controllers, the setting of neural or neural-fuzzy networks' parameters, path planning, greenhouse climate control, economic load dispatch, etc. [Whitley 91] [Wang 08] [Maield 94] [Labbi 05] [Yuan 11]. It is especially useful for complex high-dimension optimisation problems.

The classical GA is based on binary calculation, which has limitations when applying to multi-dimensional and high-precision numerical optimisation problems [Michalewicz 94]. For instance, if the problem dimension is 100, the variables are in the range between −500 to 500, and the required solution precision is 10^{-6}, the required binary solution vector is 3000 in size. As a result, the solution space consists of 2^{3000} points. It is a very wide solution space that degrades the GA performance much, especially on searching time. This drawback could be overcome by using real-coded genetic algorithm (RCGA). In RCGA, each element in chromosome is represented by a floating-point number. Thanks to the nature of floating-point numbers, a large problem domain can be handled without involving a large solution

space. Many EC algorithms have been proposed based on the idea of RCGA. One of the key successful areas is the swarm-based optimisation.

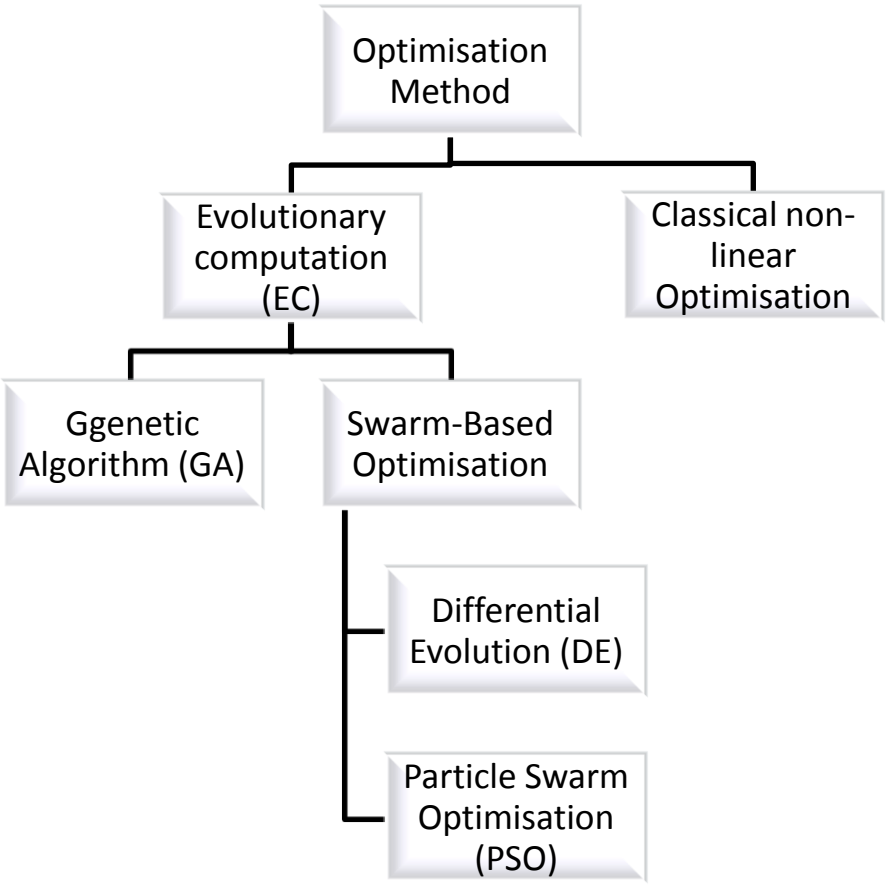


Fig. 2.1. Summary of different optimisation methods families.

I SWARM-BASED OPTIMISATION

With the success of GA, a family of evolutionary algorithms called swarm-based optimisation was introduced to improve further the performance of GA. Swarm-based optimisation is a population-based stochastic optimisation technique based on swarm intelligence, which is an artificial intelligence technique based on the study of population

behaviour. Beni and Wang first introduced it to solve the problem of cellular robotic systems [Beni 89].

In the system of swarm intelligence, a group of agents forms a population. The structure of each agent should be simple. During the optimisation, the agents share information with each other in order to process the measurement and searching [Bonabeau 99]. The way of interaction is often inspired by operations in natural and biological systems. One of the key characteristics of swarm intelligence is that there is no centralised control among the agents [Rifaie 12]. All agents are following simple rules to control their own operations. The simple rules govern individual agents to operate based on the neighbouring agents' status. We could consider these simple rules are for local information exchange. The effect of local information exchange is that every agent can communicate with other agents by means of direct or indirect connections [Kennedy 01]. Direct connection means an agent updates its status based on some selected agents. Usually they are the neighbour agents. Indirect connection means the agents update their status implicitly based on the information of other groups of agents. As a result, the groups of agents could form a global inter-connection network for information exchange. Hence, a global intelligent behaviour can be archived by swarm-based optimisation. Examples of natural rules that have been applied in swarm intelligence include fish schooling, animal herding, bacterial growth, bird flocking, and ant colonies [Dorigo 97] [Dorigo 99] [Miller 10] [Rifaie 12] [Jens 10] [Sabhin 05] [Kennedy 01] [Vicsek 08].

Two of the most successful swarm intelligence algorithms are Differential Evolution (DE) [Storn 97] and Particle Swarm Optimisation (PSO) [Kennedy 95]. Derivative information of the objective function is not required in both DE and PSO. They only involve simple mathematical operations. As a result, they can be implemented easily with simple

computer platforms, even in embedded systems with a low-end central processing unit (CPU). Moreover, they can be implemented easily with any computer language.

A. Differential Evolution

Differential Evolution (DE) has been applied in many different kinds of optimisation problem successfully. It is a stochastic optimisation algorithm, which controls a population to search the solution space for the optimal solution. The difference between two vectors in the population are used to guide the population to the optimal point. The control scheme is self-organizing because it does not require separating the probability distribution [Chakrakraborty 08] [Das 11].

DE belongs to the class of evolutionary algorithms (EAs). The operation of EAs is inspired by biological evolution. EAs work with a population to search for the optimal solution. As compared with other optimisation methods, EAs have lower chances of trapping in local optima. As a result, EAs are viewed as a tool for handling global optimisation problem by many researchers. Some of the popular examples of EAs include Evolutionary Programming (EP) and Genetic Algorithm (GA) [Goldberg 89] [Yao 99]. Like most of the methods in EAs, DE controls a population to perform evolution to guide the population to move towards the global solution.

In the operation of DE, a trial vector is generated by a simple subtraction of two other vectors in the population. As a result, exploration of the solution space is realised. As only a simple subtraction is required, DE is very easy to implement. Many applications have been applied with DE successfully, for example, optimisation of non-linear functions [Babu 01], power plant control [VanSickel 07], and data clustering [Paterini 04].

Like many EC methods, DE performs optimisation with a population. The size of the population remained unchanged until the end of searching. In this chapter, the population in

DE is denoted as $P_{x,g}$, where g is the current number of generation. The population $P_{x,g}$ is defined as follows:

$$\begin{aligned} &P_{x,g} = (\mathbf{x}_{i,g}), i = 0,1,...,N_p - 1; g = 0,1,...,g_{\max} \\ &\mathbf{x}_{i,g} = (x_{j,i,g}), j = 0,1,...,D-1. \end{aligned} \tag{2.1}$$

where $\mathbf{x}_{i,g}$ is the i-th vector in the current population; $g_{\max}$ is the maximum number of generation; the dimension of the vector is D; N_p is the total number of vectors in the population. The searching operation of DE is bounded in a special area of solution space. Before the searching process starts, the population should be distributed randomly and uniformly in the searching space. During the searching process, DE performs two major operations to control the movement of population. The first operation is called mutation, and the second operation is called crossover. For mutation, DE generates a mutated vector $\mathbf{v}_{i,g}$ by adding a scaled difference of two randomly selected vectors in the population to the target vector $\mathbf{x}_{i,g}$. The mutation operation is defined as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}) \tag{2.2}$$

where r_1 and r_2 are two random integers between 0 and N_p−1, F is a control parameter for the mutation operation. The value of F is used to control the rate of movement of the trial vectors. It is called the scaling factor of mutation operation. After the mutation operation, DE performs another process called crossover. The crossover operation generates a new trial vector $\mathbf{u}_{i,g}$ by using vector element $x_{j,i,g}$ and $v_{j,i,g}$. The operation of crossover is defined as follows:

$$\mathbf{u}_{i,g} = \left(u_{j,i,g}\right) = \begin{cases} \left(v_{j,i,g}\right) & \text{if } (rand_j(0,1) \le C_r \\ \left(x_{j,i,g}\right) & \text{otherwise.} \end{cases} \tag{2.3}$$

where $C_r \in [0, 1]$ is a user-defined value for controlling the number of elements in the mutated vector to generate a new trial vector. It is defined as the crossover rate. $rand_j(0,1)$ is a random number generator function for the j-th element that generates a value between 0 and 1. The crossover operation is designed to ensure that at least one element in the mutated vector is copied to the new trial vector. After the mutation and crossover operations, the DE performs the selection process for the new trial vectors. The new trial vector and the target vector are compared by their fitness function values. If the new trial vector offers a better fitness function value than the target vector, DE takes the trial vector into the next generation population; otherwise, the new trial vector is ignored and the target vector is kept in the population for the next generation. The selection operation is defined as follows:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \tag{2.4}$$

The fitness function is denotes as $f(\cdot)$. This selection operation enables the DE to have optimisation ability to obtain better solution. After a number of generations, DE takes the latest best vector as the final solution. Fig. 2.2 shows the pseudo code for the algorithm of standard DE (SDE).

```
begin
   Initialise the population
   while (not termination condition) do
      begin
         Mutation operation by (2.2)
         Crossover operation by (2.3)
         Evaluation of the fitness function
         Select the best vector by (2.4)
      end
end
```

Fig. 2.2. Pseudo code for standard Differential Evolution (SDE)

Apart from the above standard DE scheme, there are a number of other DE schemes. Price [Das 11] suggested a naming convention to identify different schemes of DE. The general convention used has a format of "DE/x/y/z", which contains four parts. The first part is a string to indicate the name of the method. Since all different schemes are based on the DE framework, this part is filled with the string "DE". The second part specifies the method of how to select the vector in the population. Examples include "rand" and "local-to-best". When "rand" is used, the vector is selected randomly. When "local-to-best" is used, the vector with the best fitness value will be selected. The third part is an integer telling the number of difference vectors considered for perturbation. The last part is a string used to indicate the type of crossover. There are two major types of crossover: exponential (exp) crossover and binomial (bin) crossover. Hence, the standard DE scheme is defined as DE/rand/1/bin in the literature. In the following, we outline five other schemes suggested by Price [Storn 97].

DE/rand to best/2/bin

DE/rand to best/2/bin uses the same mechanism as the standard DE scheme to perform optimisation. However, the donor vectors used for perturbing are different. Two

randomly selected vectors in the population and the vector with the best fitness value in the current generation are used as the donor vectors. The mutation operation is given by,

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_1 \cdot \left(\mathbf{x}_{i,g} - \mathbf{x}_{g,best}\right) + F_2 \cdot \left(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}\right) \tag{2.5}$$

where $\mathbf{x}_{g,best}$ is the vector with the best fitness value in the current generation. F_1 and F_2 are user defined weighting factors. We usual set $F_1 = F_2$ to reduce the number of control parameters in DE/rand to best/2/bin.

DE/best/1/bin

The mutation operation in DE/best/1/bin is given by,

$$\mathbf{v}_{i,g} = \mathbf{x}_{g,best} + F \cdot \left(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}\right) \tag{2.6}$$

F_1 is a user defined weighting factor. The vector with the best fitness value in the current generation is used to be the perturbed vector.

DE/rand/2/bin

In DE/rand/2/bin, five different vectors are involved in mutation operation. The weighted differences from four vectors are added to the perturbed vector to generate the trial vector. Moreover, the perturbed vector is randomly selected in the population. The mutation operation of DE/rand/2/bin is given by,

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_1 \cdot \left(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}\right) + F_2 \cdot \left(\mathbf{x}_{r_3,g} - \mathbf{x}_{r_4,g}\right) \tag{2.7}$$

F_1 and F_2 are user-defined weighting factors. We usually set $F_1 = F_2$ to reduce the number of control parameters.

DE/rand /1/bin with per vector dither

In DE/rand/1/bin with per vector dither, the weighting factor *F* is replaced by a dither function. It means that the weighting factor is no longer a constant value. The mutation operation is given by the following equation:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + dither \cdot \left(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}\right) \tag{2.8}$$

where

$$dither = F + rand(F,1) \tag{2.9}$$

DE/best/2/bin

In DE/best/2/bin, five different vectors are involved in the mutation operation. The weighted differences from four vectors are added to the perturbed vector to generate the trial vector. Moreover, the perturbed vector is the vector with the best fitness in the population. The mutation operation of DE/rand/2/bin is given by,

$$\mathbf{v}_{i,g} = \mathbf{x}_{g,best} + F_1 \cdot \left(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}\right) + F_2 \cdot \left(\mathbf{x}_{r_3,g} - \mathbf{x}_{r_4,g}\right) \tag{2.10}$$

F_1 and F_2 are user-defined weighting factors. We usually set $F_1 = F_2$ to reduce the number of control parameters.

B. Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a stochastic optimisation algorithm that uses a population to perform searching [Kennedy 97] [Poli 07]. The operation concept is inspired by the social behaviour of animals. It is a multi-agent searching that uses parallel techniques to simulate bird flocking [Angeline 98]. The multi-agents in PSO are called particles [Naka 03] [Bratton 07]. In the searching operation, the particles conceptually fly through the multi-dimensional searching space (bird flocking) and intend to fly to the global optimum point [Liang 06]. Each particle has its own position and velocity just like a bird in a group, which are updated in each generation along the searching process. The particles in PSO are grouped together to form a swarm. Different kinds of grouping topologies are proposed [Liang 06] [Kennedy 99] [Mendes 04] in the literature for achieving better performance in different applications [Naka 03] [Yoshida 00] [Chen 07].

In each generation of PSO, each particle's velocity and position in the swarm are calculated using the neighbourhood information and the previous generation result [Liu 07]. The best result among all the previous generations will be stored in memory as global information to guide the particles to move. The updating process goes on to guide all the particles moving to the optimum point until the end of the searching process [Kennedy 02]. Fig. 2.3 shows the pseudo code of the standard PSO (SPSO) algorithm.

```
begin
  Initialise the population
  while (not termination condition) do
     begin
          Update velocity v(t) and position of
          each particle x(t) by (2.11) and (2.12)
          respectively
             if  v(t) > v_max
                 v(t) =  v_max
             end
             if  v(t) < –v_max
                 v(t)= – v_max
             end
          Reproduce a new X(t)
     end
end
```

Fig. 2.3. Pseudo code for SPSO.

In Fig. 2.3, the value t denotes the iteration (generation) number, the t-th iteration swarm is denoted as $X(t)$. Each particle is denoted as $\mathbf{x}^p(t) \in X(t)$. Every particle contains κ elements. Each element at the t-th iteration is denoted as $x_j^p(t) \in \mathbf{x}^p(t)$ where $j = 1, 2, \ldots, \kappa$ and where $p = 1, 2, \ldots, \gamma$; the maximum number of particles in the swarm is denoted as γ. The problem dimension is equal to κ.

When the optimisation process starts, the particles are initialised randomly on the searching domain. A fitness function is used to evaluate the goodness of the particles. PSO controls the particles to move to the optimum point (i.e. minimum or maximum fitness value) by means of iteration steps. The swarm evolves in each iteration step by following the procedure as given by the pseudo code in Fig. 2.3. The position $x_j^p(t)$ and the velocity $v_j^p(t)$ are given by the following equations [Zhao 05]:

$$v_j^p(t) = k \cdot \begin{pmatrix} w \cdot v_j^p(t-1) \\ + \varphi_1 \cdot rand(0,1) \cdot \left(pbest_j^p - x_j^p(t-1)\right) \\ + \varphi_2 \cdot rand(0,1) \cdot \left(gbest_j - x_j^p(t-1)\right) \end{pmatrix} \tag{2.11}$$

$$x_j^p(t) = x_j^p(t-1) + v_j^p(t) \tag{2.12}$$

where $j = 1, 2, \ldots, \kappa$; $pbest^p = \left[pbest_1^p \quad pbest_2^p \quad ... \quad pbest_\kappa^p\right]$; $gbest = \left[gbest_1 \quad gbest_2 \quad ... \quad gbest_\kappa\right]$; *gbest* represents the best particles in the swarm; the velocity $v_j^p(t)$ represents the flying speed of the *p*-th particle; $pbest^p$ represents the previous best of the *p*-th particle. In SPSO, an inertia weight *w* is embedded in the calculation of the velocity. φ_1 and φ_2 are the acceleration constants governing the speed of the searching process. They are user-defined values for different applications. *rand*(0,1) generates a random number between 0 and 1. It is used to introduce randomness in the searching process. *k* is a constriction factor which is used to perform the stability analysis, which ensures all the particles in the swarm converge at the end of the searching process while the swarm maintains a wide exploration at the beginning of searching [Eberhart 00] [Clerc 02]. The relation governing *k*, φ_1 and φ_2 in SPSO is given by,

$$k = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{2.13}$$

where $\varphi = \varphi_1 + \varphi_2$ and $\varphi > 4$. The SPSO updates the particles moving direction by using the value of $pbest^p$ and *gbest* to avoid the same direction of movement for particles. This behaviour of SPSO can guide the particles moving toward to *pbest* and *gbest.*

To ensure a well balance between the exploitation and exploration of the searching space [Kennedy 01], the value of *w* in (2.11) is modified by (2.14) in each iteration step as

follows.

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{T} \times t \tag{2.14}$$

The current number of iteration is denoted as t and the maximum number of iteration is denoted as T. The w_{max} is the upper limits of inertia weight and the w_{min} is lower limits of the inertia weight.

From observation and experiment results, SPSO converges early in the searching process [Mo 07] [Hsieh 09]. It means that the particles can move fast to the surrounding area of the optimum. However, when the particles are near to the optimum, SPSO loses the ability to guide the particles further. Experiment results show that a large number of iteration is required to make the particles moving closer to the optimum. This behaviour of SPSO can be explained from the velocity update of (2.12). If a particle is already close to the global best, the PSO will guide that particle to move away from that area to avoid trapping in the local optimum. A phenomenon called stagnation [Eberhart 98] will happen when *gbest* cannot be updated further (with a better global result found). All the particle velocities are nearly zero. It means that the particles cannot move far away to search for a better result to update *gbest.* If this situation keeps unchanged, all the particles will move to the same position of the global best particle and stop moving. As a result, no further improvement can be obtained on the searching result and the convergence of the swarm is stopped.

The mutation operation of GA is employed in SPSO to avoid the phenomenon of stagnation [Ahmed 05]. The algorithm of SPSO with mutation operation is called APSO. The employed mutation operation randomly chooses some particles within the swarm and modifies their element values by using some control scheme. The major objective of the

mutation is to provide some additional force to move the particles along the searching to obtain better results. The mutation operation in APSO is defined as follows:

$$mut(x_j) = x_j - \omega \tag{2.15}$$

where x_j is an element of a particle which is selected randomly inside the swarm; the value of ω is randomly chosen in the range of $\left[0,\ \ 0.1\times\left(para_{\max}^{j} - para_{\min}^{j}\right)\right]$ where $para_{\min}^{j}$ and $para_{\max}^{j}$ are the lower and upper bounds of the *j*-th element in the particle respectively. Only 10% of the solution space is involved. Fig. 2.4 shows the mutation operation in the APSO pseudo code. The mutation operation of APSO is performed after position update in (2.12).

```
begin
    Initialise the population
    while (not termination condition) do
        begin
            Perform the process of PSO (shown in Fig. 2.3)
            Perform mutation operation
            Reproduce a new X(t)
        end
end
```

Fig. 2.4. Pseudo code for hybrid PSO with mutation operation.

Self-Learning Particle Swarm Optimiser (SLPSO) has been proposed to overcome the drawback of SPSO [Li 12]. In SPSO, the same strategy is used to control the movement of all particles in the swarm. It reduces the performance of SPSO on solving complex problems. In SLPSO, four strategies are employed to control the movement of particles in the swarm. To control which strategy is used, an adaptive learning system is implemented in SLPSO to perform the decision-making. As a result, the particles in the swarm can move more effectively in the solution space. HPSOM is another hybrid PSO method to overcome the drawback of SPSO [Ahmed 05]. In HPSOM, a mutation operation is introduced, which

makes the particles in the swarm to have better performance on exploring the solution space. As a result, too early convergence could be avoided.

Different successful industrial applications of PSO can be found. Examples include control system [Gaing 04], routing [Gudise 04], prediction [Yu 07], and power system [Esmin 05]. Comparable searching performance has been observed in PSO with a faster convergence rate and higher solution reliability. Yet, some literature defined PSO as a local searching algorithm only [Kennedy 01] [Kennedy 95].

II ECONOMIC LOAD DISPATCH WITH VALVE-POINT LOADING (ELD-VPL)

Nowadays, electricity supply is one key issue for modern cities. To minimise the consumption of natural resources, to reduce electricity generation cost, to provide stable electricity supply, and to meet the growth of electricity demand are all important factors for operating power supply systems. The Economic Load Dispatch (ELD) problem concerns the modelling method for consolidating those relevant factors, and formulating it as a single objective optimisation problem. Fig. 2.5 shows an overview of the ELD problem.

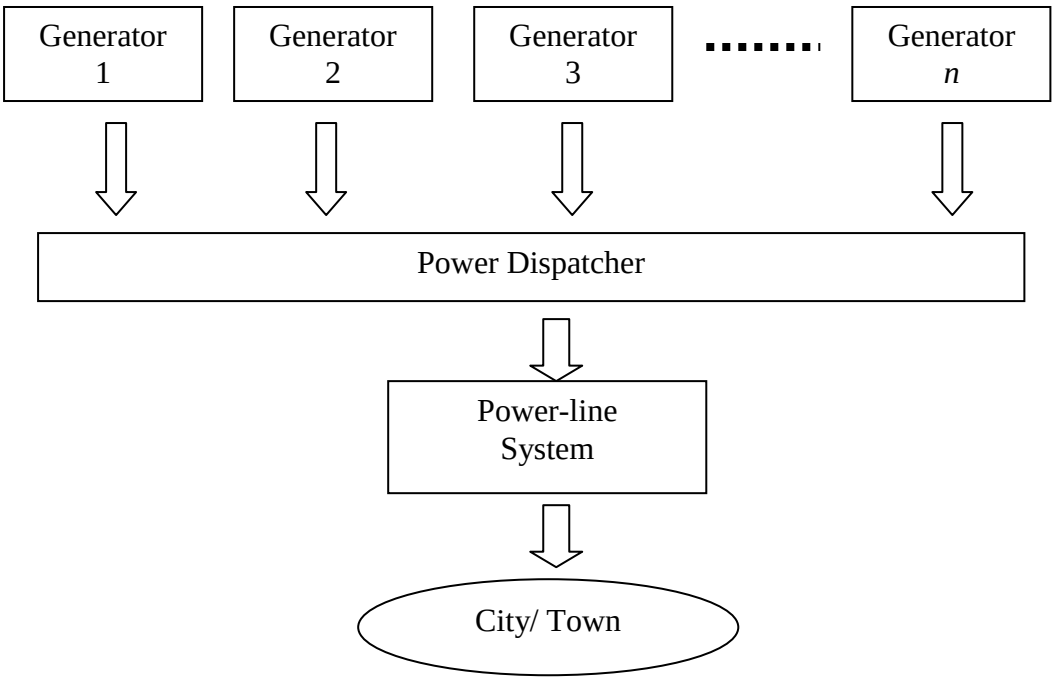


Fig. 2.5. Overview of the ELD problem.

In the ELD problem, electric power utilities (companies) are expected to maximise the profit by minimizing the operating cost on generating the power to the clients. The loading demand and transmission losses should be considered on providing a stable power supply. For secure operation, the demand of power should be dispatched to different generators correctly such that the generation capacity limits of individual generators are not exceeded. As a result, the major purpose of solving the ELD problem is to control a group of power generators to generate enough electricity with minimum fuel cost, and operates the generators within their physical constraints. The ELD problem has two characteristics that make the input-output relationship of the model become highly nonlinear by nature: the effect of the power generators' valve-point loadings in the fuel cost function and the rate limits of the generators. As a result, the objective function for the ELD with Valve-Point Loading

(ELD-VPL) problem is multimodal, highly nonlinear, and discontinuous in the solution space, high in dimension, and highly constrained.

In the past decade, different kinds of optimisation methods are introduced to tackle the ELD problem. The methods are categorised into two major families: analytic methods with classical mathematics and Computational Intelligence (CI) methods. Examples of analytic methods with classical mathematics include linear programming and nonlinear programming [Adler 77] [Bui 82]. However, analytic methods have their drawbacks. For instance, the performance is highly dependent on the starting point of the searching process. It means expert knowledge is required to determine the starting point. Moreover, analytic methods can lead to trapping in local optima easily. As a result, it is not able to handle non-linear optimisation problems effectively [Farag 95], especially those problems with piecewise linear cost approximation like the ELD-VPL problem. Because of the complex nature of the ELD-VPL problem, most of the classical techniques failed to address the ELD-VPL problem satisfactorily.

With the success of the development of CI methods, many examples of applying CI techniques on solving the ELD-VPL problem have been reported. Neural networks (NN) were applied to construct the optimiser for the ELD-VPL problem [El-Sharkawy 96] [Yalcinoz 98]. Population searching methods, such as Particle Swarm Optimisation [Park 05] [Pancholi 04], and Genetic Algorithm [Youssef 00] were also applied. Moreover, a number of hybrid methods have been developed and applied. For instance, multi-pass dynamic programming (DP), the first-order gradient method, and the conventional Lagrangian relaxation approach were proposed and presented in [Chen 01]. An algorithm called DEC-SQ was also used to tackle the problem and presented in [Coelho 06]. A hybrid method of PSO was also presented in [Victoire 04].

III HYPOGLYCAEMIC EPISODES

Low level of blood glucose is a common issue for the human body. It may happen in consequence of high level of energy consumption, action of insulin and the food ingestion. When the human body suffer from low level of blood, hypoglycaemia occurs in human body. Non-diabetic persons are not common to have hypoglycaemia [Yale 04]. The problem may be due to long-time starvation, superfluous insulin, innate problem, drugs, alcohol, insufficient hormone generation, and organ defect [DCCT 95]. Study reported that the diabetic patients have a higher chance to develop hypoglycaemia in their body if they have been treated with insulin. Another study reported that young patients with intensive glycaemia control would have a high frequency of developing hypoglycaemia [Pickup 00]. Different people may have different level of blood glucose to develop hypoglycaemia in their body. In general, maintaining above 70 mg/dL (3.9 mmol/L) for fasting glucose is considered as healthy for adults. If the blood glucose level drops below 55 mg/dL, the body may start to develop hypoglycaemia symptoms [DCCT 95]. If the blood glucose level is below 50 mg/dL (2.8 mmol/L), the body is suffering hypoglycaemic. Medical treatment is required for the patients through, for example, injection or infusion of glucagon. The human body requires maintaining a certain level of glucose in order to allow the nervous system and brain to function properly. When the central nervous systems detected the presence of hypoglycaemia, it will automatically reduce the cerebral glucose consumption (neuroglycopenic symptoms) [Maia 07]. Some symptoms can be activated before neuroglycopenic symptoms, for example, sweating, weakness, fatigue, blurry or double vision, weakness extreme hunger and headache. As a result, the patient can be aware of the presence of hypoglycaemia. If the human brain does not have enough supply of glucose, the

body may suffer neuroglycopenic symptoms, for example, loss of consciousness (coma), seizures, and confusion [Amir 07].

It is particularly dangerous if the hypoglycaemia happens at night because the patient may not be able to take immediate response. As a result, mild episodes of hypoglycaemia may become serious. Study reported that more that 50% of serious episodes of hypoglycaemia happened at evening. Even with the modest insulin elevation, serious hypoglycaemia may result due to deficient glucose counter-regulation. Moreover, the dawn phenomenon makes the handling of hypoglycaemia more difficult [Weinstein 07]. The dawn phenomenon shows that the requirement of insulin for human body decreases between midnight and 5 am. Between 5 am and 8 am, the requirement of insulin for human body increases. The technology of detecting the presence of hypoglycaemia is very demanding in the medical industry [Caduff 09] [Cho 08] [Chu 08].

Constructing a real-time detector to detect the presence of hypoglycaemia by using body signals is very important for patients. The detector can be realised by a fuzzy inference system (FIS). [Seber 08] [Wang 06] [Freedman 05] [Ling 12] [Nuryani 12]. Fig. 2.6 shows an overview of the hypoglycaemia problem.

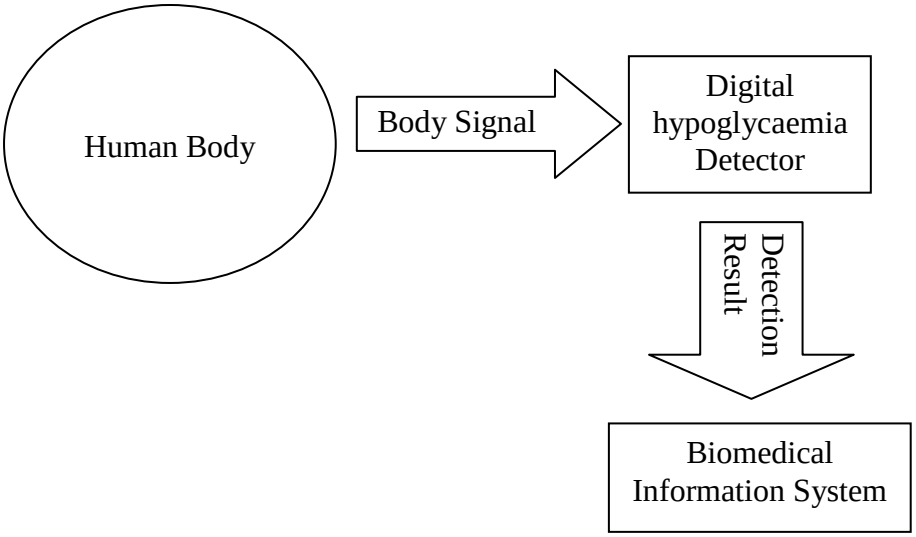


Fig. 2.6. Overview of the hypoglycaemia problem.

Different hybrid Computational Intelligence (CI) methods have been proposed to support the detecting of the presence of hypoglycaemia. Many researchers studied the real time physiological signals of mellitus (TIDM) patients and constructed detection methods for the presence of hypoglycaemia using the real-time physiological signals [Ling 11]. Different models have been proposed to detect the presence of hypoglycaemia by heart rate (HR), corrected QT interval of the electrocardiogram (ECG) signal (QT_c), change of HR, change of QT_c. In the literature, most of the developments are based on experiments using real data of TIDM patients. Several different approaches [Chu 08] [Altman 94] [Ling 11] [Chan 11] [Nuryani 12] have been used to realise the detection model. They are:

1. Fuzzy inference system
2. Linear multiple regression
3. Genetic algorithm based multiple regressions
4. Feed-forward neural network (FFNN)

5. Hybrid particle swarm optimised by fuzzy reasoning
6. Support vector machine (SVM)
7. Rule discovery system using neural network

Hypoglycaemic is a serious medical problem for young patients. A lot of study is still undergoing for constructing a reliable model to detect the blood glucose level.

Chapter 3

DIFFERENTIAL EVOLUTION WITH DOUBLE WAVELET MUTATIONS

I INTRODUCTION

In this chapter, two stages of wavelet mutation are introduced in the Differential Evolution (DE) algorithm to enhance the searching performance. The two mutations offer an effective balance between the exploration and exploitation of the solution space. As a result, better solution quality and reliability can be achieved. The first wavelet mutation is realised to replace the conventional DE mutation. A wavelet function is used to control the scaling factor in DE mutation operation. In the second stage, the DE crossover operation is added with a wavelet mutation, in which a wavelet function is used to modify the elements in the trial vectors in the population.

DE is a member of the family of Swarm Based Optimisation, which is a stochastic optimisation technique that mimics swarm intelligence in a population. In the standard DE, there are two major operations: mutation and crossover. On properly improving the mutation and crossover operations in DE, we can achieve a balance between the exploration and the exploitation of the searching space. In the early stage of searching, we want more

exploration while more exploitation is desired at the later stage of searching. Exploration means widely searching the solution space in a large area. Exploitation means searching the solution space in a small area to obtain a fine-tuned solution. This mechanism can be realised by a wavelet function [Daubechies 92]. The modulation for balancing the exploration and exploitation is realised by incorporating dilations to the oscillatory wavelet function, which take advantage of the wavelet function's properties. As a result, a more efficient searching process with better reliability in searching the global solutions of different optimisation problems can be realised. It is found that the proposed double wavelet mutations aid the DE algorithm to offer improved results for a suite of 29 benchmark test functions. The improvements are in terms of convergence rate, solution quality and solution reliability. Experimental results show that the proposed algorithm can offer better solutions over other conventional methods.

The organisation of this chapter is as follows. The proposed DE with double wavelet mutations will be presented in Section II. Section III discuss the experimental results on applying the proposed method to 29 benchmark test functions. Section IV give a conclusion for the whole chapter.

II DE WITH DOUBLE WAVELET MUTATIONS

To implement the searching operation in DE, a group of solution vectors (population) is generated population over the solution space randomly. During the searching process, the population keeps updating their position on the solution space for the optimum solution with respect to a fitness function. In ideal case, all the vectors should converge to the global optimum. In this chapter, we assume the optimum point is the point that has the smallest fitness value in the solution space to simplify the discussion. Fig. 3.1 shows the pseudo code

for the algorithm of standard DE (SDE). In this chapter, double wavelet mutations are added to the SDE algorithm to enhance the searching performance. The resulting algorithm is called DE with double wavelet mutations (DWM-DE). Fig. 3.2 shows the pseudo code of DWM-DE. The discussion of SDE and DWM-DE are discussed as follows.

```
begin
   Initialise the population
   While (not termination condition) do
      begin
         Mutation operation by equation (3.2)
         Crossover operation by equation (3.3)
         Evaluation of the fitness function
         Select the best vector by equation (3.4)
       end
end
```

Fig. 3.1. Pseudo code for standard DE (SDE).

```
begin
   Initialise the population
   While (not termination condition) do
      begin
        Update the new value of F by equation
        (3.12)
        Mutation operation by equation (3.2)
        Crossover operation by equation (3.3)
        Modifying the trial population vectors
        based on equation (3.15)
        Evaluation of the fitness function
        Select the best vector by equation (3.4)
       end
end
```

Fig. 3.2. Pseudo code for the proposed DWM-DE.

A. Standard Differential Evolution (SDE)

In each evolution, a population of N_p vectors is controlled by the DE mutation and crossover operation. Let the population for the current generation g of DE be $P_{x,g}$, and the i-th vector in this population be $\mathbf{x}_{i,g}$. As a result, we have:

$$P_{x,g} = (\mathbf{x}_{i,g}), i = 0,1,...,N_p - 1; g = 0,1,...,g_{\max}$$
$$\mathbf{x}_{i,g} = (x_{j,i,g}), j = 0,1,...,D-1. \quad (3.1)$$

where D is the number of elements in each vector and $g_{\max}$ is the maximum number of generation. During the initialisation, the population is randomly and uniformly distributed on the solution space and located within the specified boundary. After the initialisation, DE performs mutation operation for each vector in the population. The mutation operation creates a mutated vector $\mathbf{v}_{i,g}$ for each target vector $\mathbf{x}_{i,g}$. The operation of mutation is defined as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}) \quad (3.2)$$

where r_1 and r_2 are two random integers between 0 and N_p–1, F is a control parameter for the mutation operation. The value of F is used to control rate of movement of the trial vectors. It is called the scaling factor of mutation operation. After the mutation operation, DE performs another operation called crossover. The crossover operation generates a new trial vector $\mathbf{u}_{i,g}$ by using vector element $x_{j,i,g}$ and $v_{j,i,g}$. The operation of crossover is defined as follows:

$$\mathbf{u}_{i,g} = (u_{j,i,g}) = \begin{cases} (v_{j,i,g}) & \text{if } (rand_j(0,1) \le C_r \\ (x_{j,i,g}) & \text{otherwise.} \end{cases} \quad (3.3)$$

where $C_r \in [0, 1]$ is a user-defined value which is used to control the number of mutated elements in the new trial vector. It is called the crossover rate. $rand_j(0,1)$ is a random number generator function for the j-th element, which is used to generate a value between 0 and 1. The crossover operation is designed to ensure that at least one element in the mutated

vector is copied to the new trial vector. After the mutation and crossover operations, the DE performs the selection process for the new trial vectors. The new trial vector and the target vector are compared by means of the fitness function value. If the new trial vector offers a better fitness function value that the target vector, DE takes the trial vector into the next generation population; otherwise, the new trial vector is ignored and the target vector is kept in the population for the next generation. The selection operation is defined as follows:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \tag{3.4}$$

The fitness function is denotes as $f(\cdot)$. This selection operation offers the DE to have optimisation ability for a better solution. After a number of generations, DE takes the latest population as the final solution.

B. Differential Evolution with Double Wavelet Mutations (DWM-DE)

To apply SDE to an optimisation problem, the user is required to determine the value of the scaling factor *F*. This value is a fixed value in SDE and it is commonly set within the range of [0, 1]. Different applications require different values of the scaling factor *F* to obtain good performance. The user needs to rely on their experience and expert knowledge to determine this value. Moreover, a fixed value may not be able to provide a good balance between the exploration and exploitation of the searching space. As a result, we propose the value of *F* keeps diminishing with the increase of the number of iteration. This leads to the idea on controlling the value *F* by a wavelet function. By using the wavelet function to control the value *F*, different degrees of movement of the population is achieved. As a result,

a high degree of exploration for searching can be realised in the early stage. Controlling the value of *F* by a wavelet function is the first stage of wavelet mutation in the proposed DWM-DE. For the second stage of wavelet mutation, it is realised in the crossover operation in which a wavelet-based second mutation mechanism is embedded to modify the trial vectors within the population. The searching performance on exploration and exploitation can be enhanced. In the two stages of mutations, the output of wavelet function is configured to be continuous decreasing alone the searching process. It means than the output is inversely proportional to the number of iteration. Therefore, the effect of the wavelet mutation is reduced at the later stage of searching. Exploitation can be achieved when the effect of the wavelet mutation is reduced. As a result, the good balance between the exploration and exploitation can be realised, and less number of iteration is required for obtaining the best solution. Moreover, by taking advantage of the wavelet function's properties, the solution reliability is increased in a statistical sense. The solution reliability in the DE algorithm is an important factor to evaluate the searching performance due to the random factor present in the searching operation.

C. Double Wavelet Mutations

1. Wavelet function

A wavelet function is used to model seismic signals. Modulation can be realised by incorporating translations and dilations to the oscillatory function. The function is bounded within a finite domain. It satisfies the following properties:

Property 1:

$$\int_{-\infty}^{+\infty} \psi(x)dx = 0 \tag{3.5}$$

$\psi(x)$ is a continuous-time function. It means that the momentum of positive side and negative side are equal in terms of the area of $\psi(x)$.

Property 2:

$$\int_{-\infty}^{+\infty}|\psi(x)|^2 dx < \infty \tag{3.6}$$

It means that most of the energy is bounded in a finite domain. Equation (3.7) and Fig. 3.3 show an example mother wavelet function.

$$\psi(x) = e^{-x^2/2}\cos(5x) \tag{3.7}$$

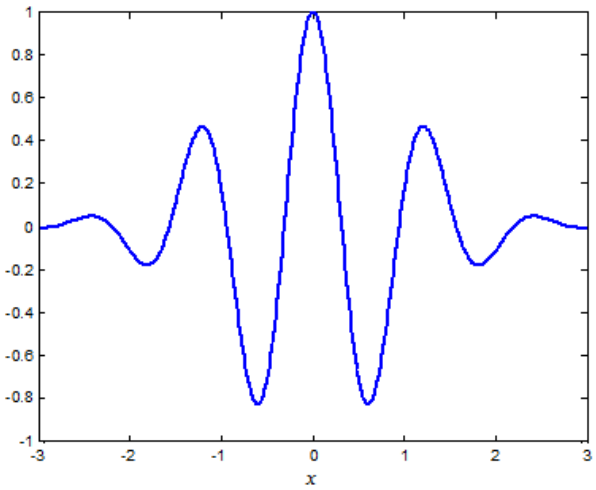


Fig. 3.3. Morlet wavelet.

According to property 1, the area of positive side and negative side of the Morlet wavelet function are equal. In addition, according to property 2, the interval $-2.5 < x < 2.5$ of the function contains most of function energy (>99%). To control the magnitude of $\psi(x)$, a dilation parameter is introduced, which is denoted by a. The amplitude-scaled $\psi(x)$ is defined as follows.

$$\psi_a(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right) \tag{3.8}$$

Different dilations of the Morlet wavelet is shown in Fig. 3.4. The amplitude of $\psi_a(x)$ is controlled by the dilation parameter *a*. The output of $\psi_a(x)$ is proportional to the value of a.

2. Operation of DE wavelet mutation

The operation of proposed wavelet mutation (WM) is defined as follows.

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot \left(\left| \mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g} \right| \right), \tag{3.9}$$

where

$$F = \psi_a(\varphi), \tag{3.10}$$

$$F = \frac{1}{\sqrt{a}} \psi\left(\frac{\varphi}{a} \right). \tag{3.11}$$

Using (3.7) as the mother wavelet, the value of *F* is given as follows:

$$F = \frac{1}{\sqrt{a}} e^{-\left(\frac{\varphi}{a}\right)^2 / 2} \cos\left(5\left(\frac{\varphi}{a} \right) \right), \tag{3.12}$$

With the property of wavelet function defined in (3.5), the momentum of positive side and negative side are equal in terms of the area for the mother wavelet. With *N* samples, if the value of *N* is large and the value of φ is chosen randomly, the sum of the value *F* along the evolution is equal to zero, i.e.

$$\frac{1}{N} \sum_{N} F = 0 \text{ for } N \to \infty, \tag{3.13}$$

The solution reliability can achieved by Property 1 and 2 as defined in (3.5) and (3.6). Better solution reliability means that the algorithms can offer very similar solution for many trials. The solution reliability can be measured by determining the standard deviation of the solution. If the standard deviation of the solution is small, it means that the solution reliability is good. As most of the energy is bounded in a finite domain for a wavelet function, we could control the output amplitude by using the dilation parameter *a* in order to control the searching for exploration or exploitation. In the proposed wavelet mutation, the value of *a* is defined as follows, which introduces a monotonic increasing behaviour to the wavelet function:

$$a = e^{-\ln(\lambda)\times\left(1-\frac{t}{T}\right)^{\zeta_{wm}}+\ln(\lambda)} \tag{3.14}$$

where the current number of iteration in the evolution is denoted as *t*, the total number of iteration is denoted as *T*, λ is used to limit the maximum value of *a* and ζ_{wm} is the shape parameter of the monotonic increasing function. From equation (3.14), the value of *a* vary with the value of *t*. If the value of *t* is large, the effect of mutation is decreased.

Fig. 3.5 shows the of the shape parameter ζ_{wm} to *a* with respect to *t/T*. In this figure, the value of *a* is between 1 and 10000 (for λ = 10000). When the value *t*, φ are equal to zero and the value of *a* is equal to 1, equation (3.12) gives the maximum value of *F*. When the value *t* is nearly equal to *T* and the value of *a* is large, equation (3.12) gives the minimum value of *F*. As a result, the wavelet mutation could offer more exploration in the early stage of searching while more exploitation is done at the later stage.

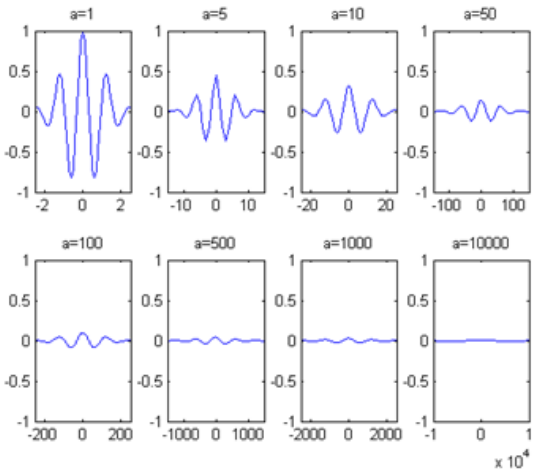


Fig. 3.4. Morlet wavelet dilated by different values of a (x-axis: a, y-axis: $\psi_a(x)$.)

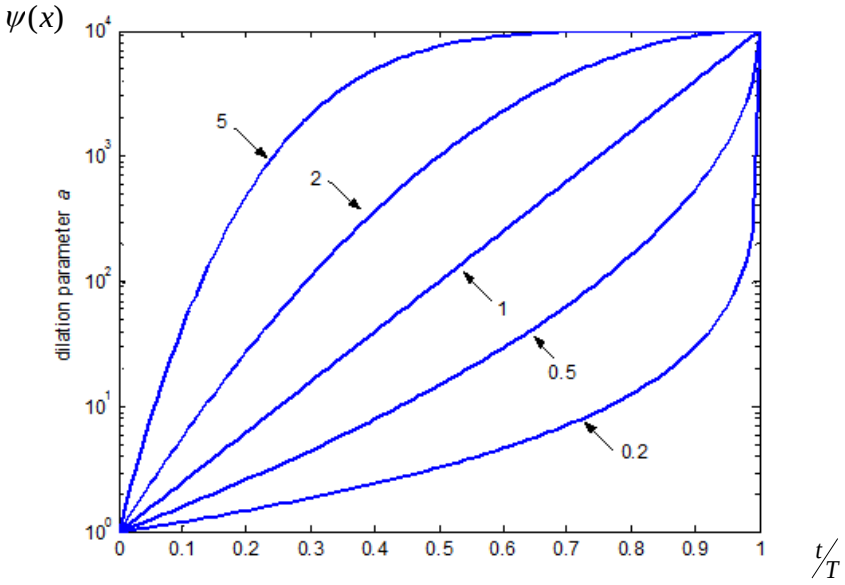


Fig. 3.5. Effect of the shape parameter ζ_{wm} to a with respect to t/T.

3. Operation of DE crossover with wavelet mutation

In the proposed DWM-DE, the DE crossover operation is added with a second stage of wavelet mutation [Neubauer 97]. The second stage wavelet operation is done after the SDE crossover operation to offer a balance between the exploration and exploitation of the searching space. The resulting i-th vector of the crossover operation is denoted as

$\mathbf{u}_{i,g} = \left(u_{0,i,g}, u_{1,i,g}, \ldots, u_{D-1,i,g}\right)$ where i is the vector number of the population, g is the current generation number, and the maximum number of elements in the vector is denoted as D. Let $\bar{\mathbf{u}}_{i,g} = \left(\bar{u}_{0,i,g}, \bar{u}_{1,i,g}, \ldots, \bar{u}_{D-1,i,g}\right)$ be the mutated crossover vector and

$$\bar{u}_{j,i,g} = \begin{cases} u_{j,i,g} + \sigma \times \left(para^{j}_{\max} - u_{j,i,g}\right) \text{ if } \sigma > 0 \\ u_{j,i,g} + \sigma \times \left(u_{j,i,g} - para^{j}_{\min}\right) \text{ if } \sigma \le 0 \end{cases}, \tag{3.15}$$

$$\sigma = \psi_a(\varphi) = \frac{1}{\sqrt{a}} \psi\left(\frac{\varphi}{a}\right) \tag{3.16}$$

where $\left[para^{j}_{\min}, para^{j}_{\max}\right]$ is the boundary of each element in the solution vector. Equation (3.14) is used to control the value of a. Following the same behaviour of F in equation (3.12), in the early stage of searching, we want to explore more in the searching space. As a result, we want the value of $|\sigma|$ to be large. In addition, we want more exploitation in the searching space at the later stage of searching. As a result, we want the value of $|\sigma|$ be small. After the second wavelet mutation, the DWM-DE used the same method of standard DE to perform the population selection process.

III. BENCHMARK TEST FUNCTIONS AND RESULTS

A. Benchmark Test Functions

A suite of 29 benchmark test functions [Brest 06] [Ao 09] [Fan 03] [Ali 05] is used to evaluate the performance of the proposed DWM-DE. These 29 benchmark test functions covers many different kinds of optimisation problems and can be separated into four main categories. The functions f_1 - f_8 (first category) are unimodal functions that involve a

symmetric solution space and contain a single optimum point only. The functions f_9 - f_{16} are multimodal functions with a few local minima; they are put to the second category. The third category covers the multimodal functions with many local minima; functions f_{17} - f_{24} belong to this category. The last category contains functions with shift and rotate; functions f_{25} - f_{29} belong to this category, which are the shifted and rotated versions of some functions of the pervious categories.

Table 3.1 shows the names of the test functions. The definitions of these functions are listed in Table 3.2. The definitions of function parameters a, b, c and m, and the function $u(\cdot)$ are provided in [Ali 05].

Table 3.1. Benchmark Test Functions Family.

Category of Test Functions	**Names of Test Functions**
Unimodal functions	f_1. Sphere model
	f_2. Generalised Rosenbrock's function
	f_3. Step function
	f_4. Quartic function
	f_5. Schwefel's problem 2.21
	f_6. Schwefel's problem 2.22
	f_7. Easom's function
	f_8. McCormick function
Multimodal functions with a few local minima	f_9. Shekel's foxholes function
	f_{10}. Kowalik's function
	f_{11}. Maxican hat function
	f_{12}. Six-hump camel back function
	f_{13}. Hartman's family 1
	f_{14}. Hartman's family 2
	f_{15}. Egg Holder function
	f_{16}. Styblinski-Tang function
Multimodal functions with many local minima	f_{17}. Generalised penalised's function
	f_{18}. Generalised Rastrigin's function
	f_{19}. Generalised Griewank's function
	f_{20}. Ackley's function
	f_{21}. Schwefel's function
	f_{22}. Schaffer function
	f_{23}. Chichinadze function
	f_{24}. Sine envelope sine wave function
Functions with shift and rotate	f_{25}. Shifted sphere model
	f_{26}. Shifted Schwefel's problem 1.2
	f_{27}. Shifted rotated high conditioned elliptic function
	f_{28}. Shifted Rosenbrock's function
	f_{29}. Shifted Rastrigin's function

Table 3.2. Benchmark Test Functions.

Test function	Domain	Optimal point
$f_1(\mathbf{x})=\sum_{i=1}^{30} x_i^2$	$-50\le x_i\le 150$	$f_1(\mathbf{0})=0$
$f_2(\mathbf{x})=\sum_{i=1}^{29}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	$-2.048\le x_i\le 2.048$	$f_2(\mathbf{1})=0$
$f_3(\mathbf{x})=\sum_{i=1}^{30}\left(\lfloor x_i+0.5\rfloor\right)^2$	$-5\le x_i\le 10$	$f_3(\mathbf{0})=0$
$f_4(\mathbf{x})=\sum_{i=1}^{30} ix_i^4+random[0,1)$	$-1.28\le x_i\le 2.56$	$f_4(\mathbf{0})=0$
$f_5(\mathbf{x})=\max_i\left\{\lvert x_i\rvert, 1\le i\le 30\right\}$	$-150\le x_i\le 50$	$f_5(\mathbf{0})=0$
$f_6(\mathbf{x})=\sum_{i=1}^{30}\lvert x_i\rvert+\prod_{i=1}^{30}\lvert x_i\rvert$	$-5\le x_i\le 15$	$f_6(\mathbf{0})=0$
$f_7(\mathbf{x})=-\cos\left(x_1\right)\cdot\cos\left(x_2\right)\cdot\exp\left(-\left(\left(x_1-\pi\right)^2+\left(x_2-\pi\right)^2\right)\right)$	$-300\le x_1,x_2\le 300$	$f_7([\pi,\pi])=-1$
$f_8(\mathbf{x})=\sin(x_1+x_2)+(x_1-x_2)^2-1.5x_1+2.5x_2+1$	$-1.5,-3\le x_1,x_2\le 4,4$	$f_8([-0.54719,-1.54719])=-1.9133$
$f_9(\mathbf{x})=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right]$	$-65536\le x_i\le 65536$	$f_9([-32,-32])\approx 1$
$f_{10}(\mathbf{x})=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_ix_2\right)}{b_i^2+b_ix_3+x_4}\right]^2$	$-5\le x_i\le 5$	$f_{10}([0.1928,0.1928,0.1231,$ $0.1358)\approx 0.0003075$
$f_{11}(\mathbf{x})=-\frac{\sin\left(x_1\right)\sin\left(x_2\right)}{x_1x_2}$	$-5\le x_1,x_2\le 15$	$\lim_{\mathbf{x}\to[0,0]} f_{11}(\mathbf{x})=-1$
$f_{12}(\mathbf{x})=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	$-5\le x_1,x_2\le 5$	$f_{12}([-0.08983,0.7126])$ ≈ -1.0316
$f_{13}(\mathbf{x})=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right]$	$0\le x_i\le 1$	$f_{13}([0.114,0.556,0.853])$ ≈ -3.8628
$f_{14}(\mathbf{x})=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right]$	$0\le x_i\le 1$	$f_{14}([0.201,0.15,0.477,$ $0.275,0.311,0.627])\approx -3.32$
$f_{15}(\mathbf{x})=\sum_{i=1}^{m-1}\left[\begin{matrix}(x_{i+1}+47)\sin\left(\sqrt{\left\lvert x_{i+1}+\frac{x_i}{2}+47\right\rvert}\right)+\\ \sin\left(\sqrt{\lvert x_i-(x_{i+1}+47)\rvert}\right)(-x_i)\end{matrix}\right]$	$-512\le x_i\le 512$	For m = 2, $f_{15}([512,404.2319])\approx 959.64$
$f_{16}(\mathbf{x})=\frac{1}{2}\sum_{i=1}^{2}\left(x_i^4-16x_i^2+5x_i\right)$	$-5\le x_1,x_2\le 5$	$f_{16}([-2.903534,-2.903534)$ ≈ -78.332
$f_{17}(\mathbf{x})=0.1\left\{\begin{matrix}\sin^2\left(\pi 3x_1\right)\\ +\sum_{i=1}^{29}(x_i-1)^2\cdot[1+\sin^2(3\pi x_{i+1})]\\ +(x_{30}-1)^2[1+\sin^2(2\pi x_{30})]\end{matrix}\right\}+\sum_{i=1}^{30}u\left(x_i,5,100,4\right)$	$-50\le x_i\le 50$	$f_{17}(\mathbf{1})=0$
$f_{18}(\mathbf{x})=\sum_{i=1}^{30}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	$-5.12\le x_i\le 10.24$	$f_{18}(\mathbf{0})=0$
$f_{19}(\mathbf{x})=\frac{1}{4000}\sum_{i=1}^{30}x_i^2-\prod_{i=1}^{30}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$-1206\le x_i\le 600$	$f_{19}(\mathbf{0})=0$
$f_{20}(\mathbf{x})=-20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30}x_i^2}\right)-\exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos 2\pi x_i\right)+20+e$	$-64\le x_i\le 32,$	$f_{20}(\mathbf{0})=0$
$f_{21}(\mathbf{x})=\sum_{i=1}^{30}(x_i\sin(\sqrt{\lvert x_i\rvert}))$	$-500\le x_i\le 500$	$f_{21}\left([420.9687,\cdots 420.9687]\right)$ $=-12569.5$
$f_{22}(\mathbf{x})=0.5+\frac{\sin\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{\left[1+0.001\left(x_1^2+x_2^2\right)\right]^2}$	$-100\le x_1,x_2\le 100$	$f_{22}(\mathbf{0})=0$

$f_{23}(\mathbf{x}) = x - 12x + 11 + 10\cos\left(\frac{\pi x_1}{2}\right) + 8\sin(5\pi x_1) - \left(\frac{1}{5}\right)^{0.5} e^{-0.5(x_2-0.5)^2}$	$-30 \le x_1, x_2 \le 30$	$f_{23}([5.901330.5]) \approx -43.3159$
$f_{24}(\mathbf{x}) = \sum_{i=1}^{m-1}\left(\frac{\sin\left(\sqrt{x_{i+1}^2+x_i^2}\right)-0.5}{\left[0.001\left(x_{i+1}^2+x_i^2\right)+1\right]^2}\right)+0.5$	$-100 \le x_i \le 100$	$f_{24}(\mathbf{0}) = 0$
$f_{25}(\mathbf{x}) = \sum_{i=1}^{30} x_i^2$	$-100 \le x_i \le 100$	$f_{25}(\mathbf{0}) = 0$
$f_{26}(\mathbf{x}) = \sum_{k=1}^{30}\sum_{i=1}^{k} x_i^2$	$-100 \le x_i \le 100$	$f_{26}(\mathbf{0}) = 0$
$f_{27}(\mathbf{x}) = \sum_{i=1}^{30}(A^{\frac{i-1}{9}}(x_i^2)), where Ahe \times IO^{6}$	$-100 \le x_i \le 100$	$f_{27}(\mathbf{0}) = 0$
$f_{28}(\mathbf{x}) = \sum_{i=1}^{30}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	$-100 \le x_i \le 100$	$f_{28}(\mathbf{0}) = 0$
$f_{29}(\mathbf{x}) = \sum_{i=1}^{30}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$-5 \le x_i \le 5$	$f_{29}(\mathbf{0}) = 0$

B. Experimental Setup

We evaluate the performance of SDE [Chakrakraborty 08], DE with single wavelet mutation (first-stage mutation only, SWM-DE), DE/local-to-best/1 [Ao 09], DE/rand/1 with per-vector-dither [Rahnamayan 08], and the proposed DWM-DE by observing the estimated optimum values of the benchmark test functions. It this experiment, the optimum value is the minimum value of the solution space. The experimental conditions are defined as follows:

- Shape parameter of the wavelet mutation (ζ_{wm}) (for DWM-DE): 1
- Parameter λ for the monotonic increasing function (for DWM-DE): 10000.
- Initial population: Generated randomly and uniformly.
- Crossover rate (C_r): 0.5.
- Constant mutation factor (F) (for SDE, DE/local-to-best/1 and DE/rand/1 with per-vector-dither): 0.5.
- The size of population: 30.
- Numbers of iteration for all algorithms: As listed in Table 3.3.

Table 3.3. Number of Iteration.

Test Function	No. of Iteration
f_1. Sphere model	300
f_2. Generalised Rosenbrock's function	500
f_3. Step function	100
f_4. Quartic function	200
f_5. Schwefel's problem 2.21	500
f_6. Schwefel's problem 2.22	200
f_7. Easom's function	200
f_8. McCormick function	50
f_9. Shekel's foxholes function	50
f_{10}. Kowalik's function	100
f_{11}. Maxican hat function	50
f_{12}. Six-hump camel back function	50
f_{13}. Hartman's family 1	50
f_{14}. Hartman's family 2	100
f_{15}. Egg holder function	1000
f_{16}. Styblinski-Tang function	50
f_{17}. Generalised penalised's function	200
f_{18}. Generalised Rastrigin's function	1000
f_{19}. Generalised Griewank's function	200
f_{20}. Ackley's function	500
f_{21}. Schwefel's function	500
f_{22}. Schaffer function	1000
f_{23}. Chichinadze function	50
f_{24}. Sine envelope sine wave function	2000
f_{25}. Shifted sphere model	500
f_{26}. Shifted Schwefel's problem 1.2	500
f_{27}. Shifted rotated high conditioned elliptic function	200
f_{28}. Shifted Rosenbrock's function	200
f_{29}. Shifted Rastrigin's function	1000

C. Results and Discussion

In this experiment, all results reported for the benchmark test functions are averaged ones out of 50 trials.

1. Unimodal functions

Function f_1 has a smooth and symmetric surface around the solution. It is a model of sphere. Due to its smooth surface, most of the methods can converge to the global minimum but at different rates. Fig. 3.6a shows the convergence rates. It shows that the proposed DWM-DE could provide the best performance in terms of convergence rate. In terms of mean cost value and best cost value, it offers better result than the other methods as shown in Table 3.4. In addition, the standard deviation is small, which means that the proposed DWM-DE offer a reliable searching mechanism.

Function f_2 is the Generalised Rosenbrock's function. It has a smooth and symmetric surface around the solution. The function shape looks like a "Banana". The solution space of this function contains a flat gorge. Similar to f_1, the main purpose of testing is to measure the convergence rate of the searching methods. The result is shown in Fig. 3.6b. The proposed DWM-DE offers the highest convergence rate. When using the proposed DWM-DE, the solution quality is increased. In terms of mean value and standard deviation as shown in Table 3.4, DWM-DE performs better than the other methods.

Function f_3 is a Step function. Most of the optimisation algorithms failed to handle the functions with flat surface because the optimisation algorithms could not obtains any information of searching direction from the flat surface. Yet, all the methods involved in this study could handle this function well as shown in Fig. 3.6c and Table 3.4.

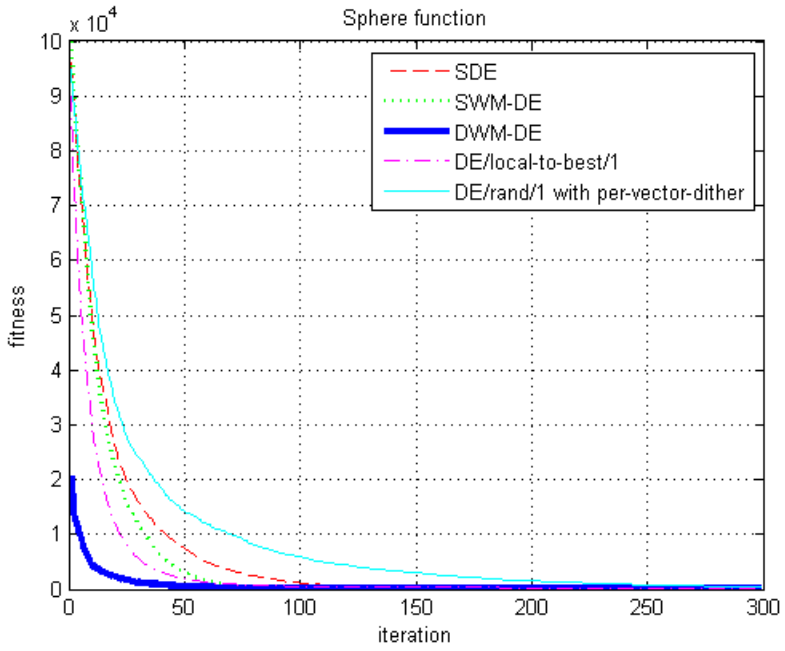
Function f_4 is the Quartic function. It contains a global minimum at the centre of the solution space. The experiment results are shown in Fig. 3.6d and Table 3.4. The

convergence rate offered by the proposed DWM-DE is much higher than that of the other DE methods. After around 10 times of iteration, the proposed DWM-DE is able to reach the minimum.

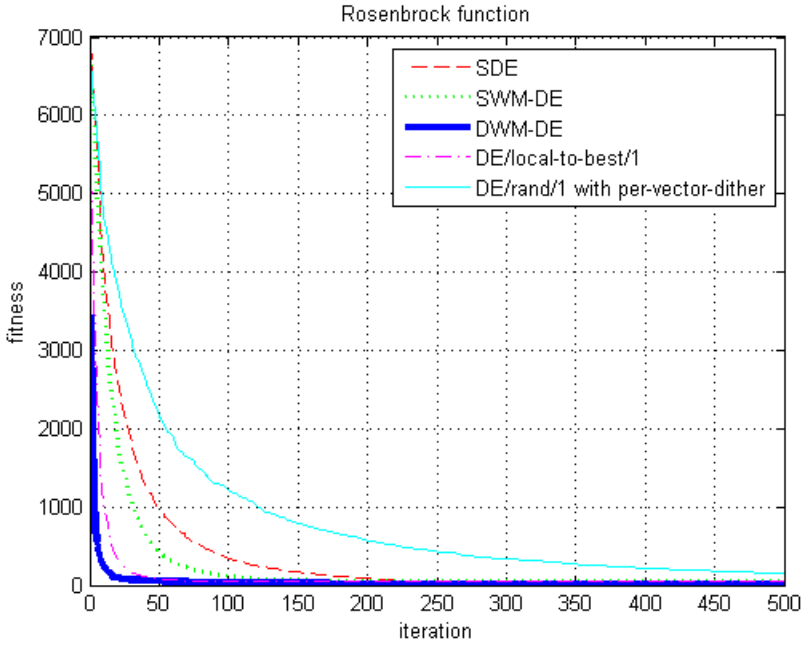
Functions f_5 is the Schwefel's problem 2.21. Fig. 3.6(e) shows the experiment results. Functions f_6 is the Schwefel's problem 2.22. The experiment results are shown in Fig. 3.6(f). The convergence rate for functions f_5 and f_6 of the proposed DWM-DE is the highest. The best solution, mean and standard derivation provided by DWM-DE are the best as shown in Table 3.4. Thus, the proposed algorithm gives better solution quality and reliability.

Function f_7 is the Easom function. The function was inverted for minimisation. A very large solution space for this function is designed to test the performance of the proposed DWM-DE. The result is shown in Fig. 3.6g. The convergence rate for functions f_7 of the proposed DWM-DE is the highest. It gives better performance in terms of solution quality and reliability as shown in Table 3.4.

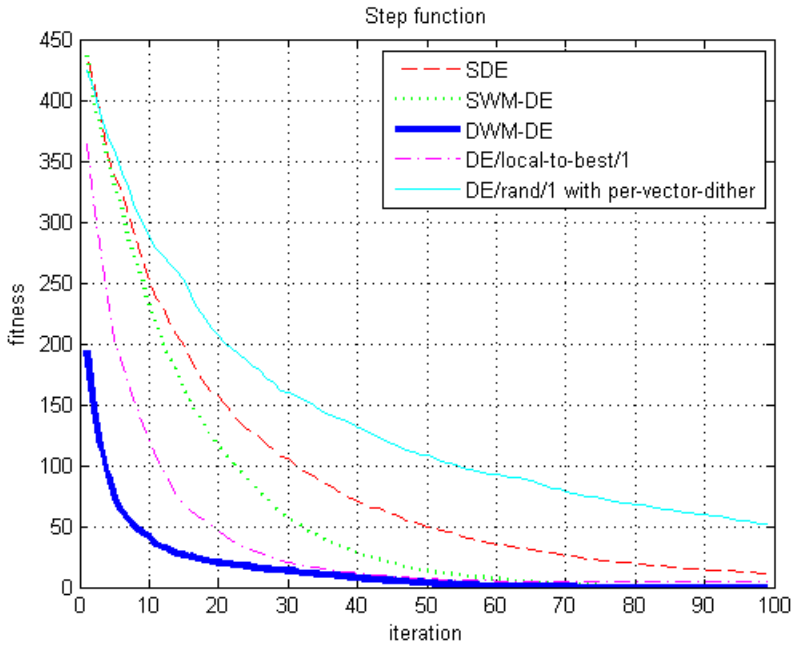
Function f_8 is the McCormick function, which is a two-dimension benchmark function with the global minimum at $f(-0.54719,-1.54719) = -1.9133$. The global minimum is not located at the centre of the search domain. The McCormick function contains a flat and smooth surface. The experiment result shows that DWM-DE can provide the best solution quality and the best solution reliability.


Sphere function
x 10^4
fitness
iteration
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

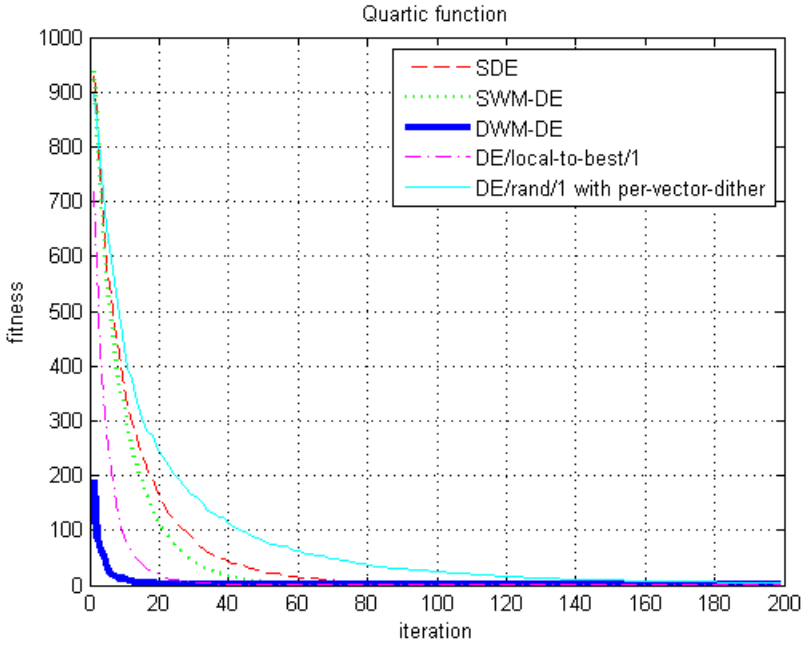
(a)


Rosenbrock function
fitness
iteration
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

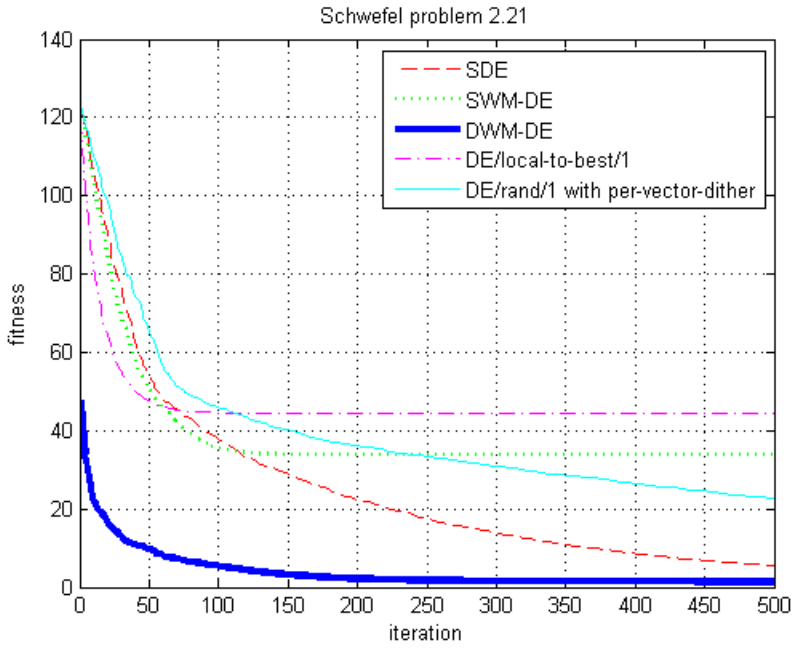
(b)


Step function
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
50
100
150
200
250
300
350
400
450
0
10
20
30
40
50
60
70
80
90
100

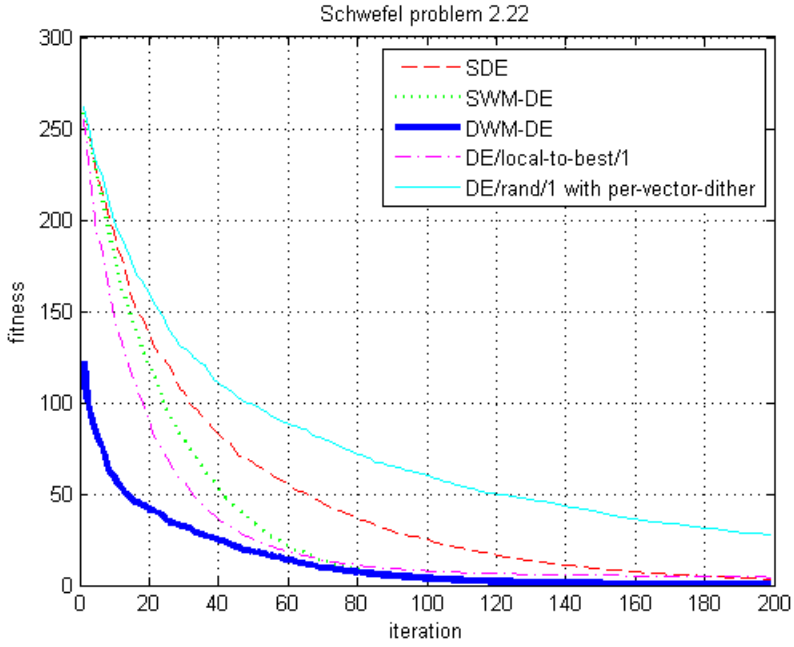
(c)


Quartic function
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
100
200
300
400
500
600
700
800
900
1000
0
20
40
60
80
100
120
140
160
180
200

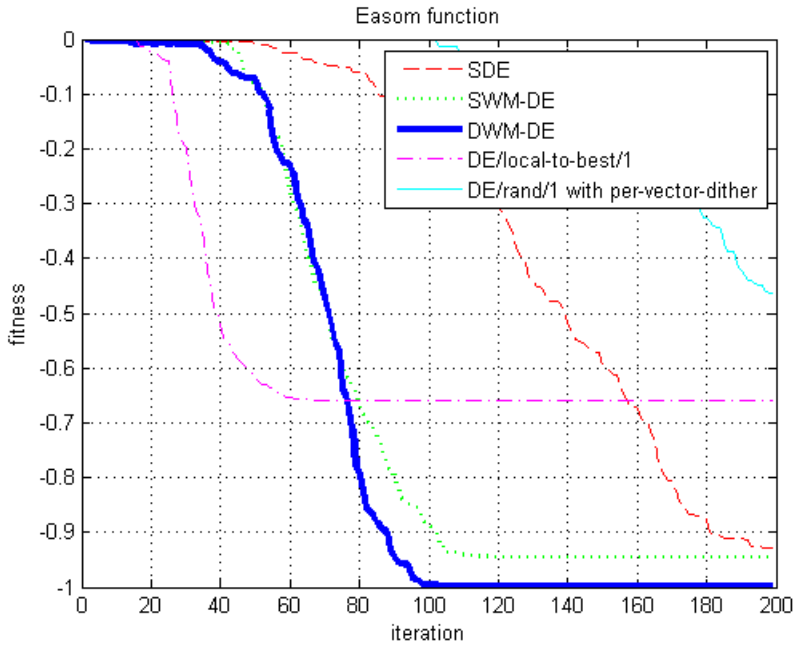
(d)


Schwefel problem 2.21
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
20
40
60
80
100
120
140
0
50
100
150
200
250
300
350
400
450
500

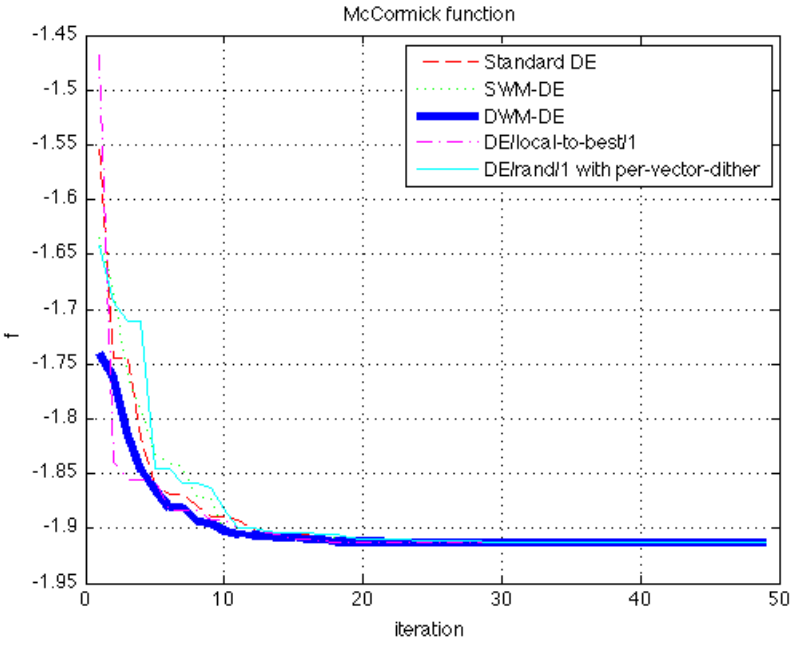
(e)


Schwefel problem 2.22
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
50
100
150
200
250
300
0
20
40
60
80
100
120
140
160
180
200

(f)



(g)



(h)

Fig. 3.6. Unimodal functions.

Table 3.4. Comparison between Different DE Methods for Benchmark Test Functions (Category 1).

		DWM-DE	SW-DE	SDE	DE/ local-to-best/ 1	DE/rand/1 with per-vector -dither
f_1	Mean	0.5902	33.9672	0.9937	228.8271	411.8185
	Best	0.0605	0.668	0.4317	17.3954	206.3942
	Std Dev	0.5712	75.9703	0.308	213.2461	99.2895
f_2	Mean	0.0961	51.5008	25.3632	40.3851	30.2437
	Best	0.0068	22.9713	23.7534	26.9933	27.3151
	Std Dev	0.0867	30.3582	0.6442	15.6161	3.9049
f_3	Mean	0	0.78	11.24	4.7	51.54
	Best	0	0	7	1	35
	Std Dev	0	0.9957	1.9119	3.4241	8.1346
f_4	Mean	0.0385	0.2103	0.2307	0.5176	4.2939
	Best	0.0172	0.0366	0.1033	0.0798	1.7894
	Std Dev	0.0111	0.2053	0.0684	0.3172	1.3556
f_5	Mean	1.4127	33.9335	5.5862	44.3662	22.7523
	Best	0.7089	16.0324	3.7069	21.1518	19.1513
	Std Dev	0.3512	9.5589	2.9702	9.0817	2.05
f_6	Mean	0.388	0.7726	3.3391	4.3577	27.5979
	Best	0.1151	0.182	2.3578	0.1878	20.7384
	Std Dev	0.187	0.7565	0.5352	3.6601	3.2958
f_7	Mean	-1	-0.9455	-0.9284	-0.66	-0.4641
	Best	-1	-1	-1	-1	-1
	Std Dev	0	0.2131	0.2488	0.4785	0.4454
f_8	Mean	-1.913223	-1.913209	-1.913199	-1.913223	-1.913152
	Best	-1.913223	-1.913223	-1.91322	-1.913223	-1.913219
	Std Dev	0	0.000018	0.000023	0	0.00007

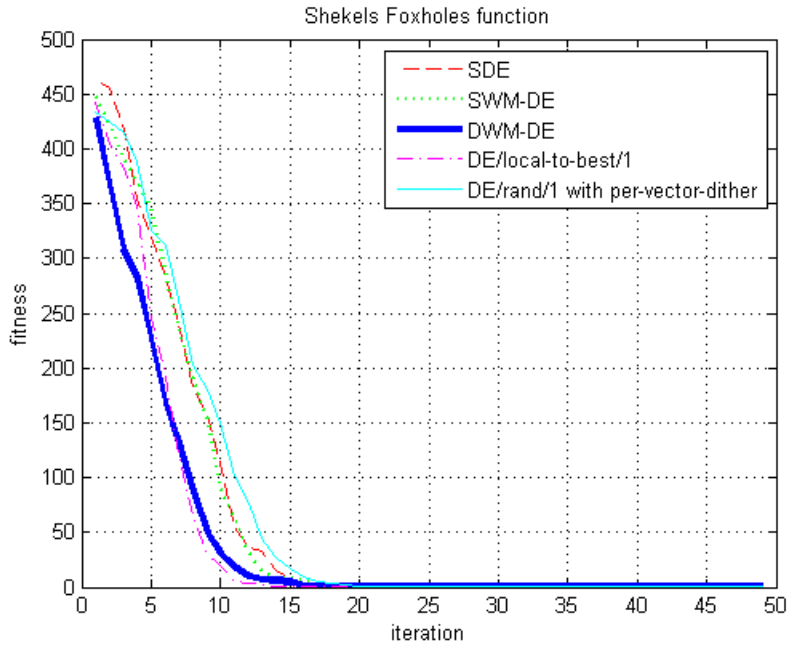
The proposed DWM-DE can offer a higher rate of convergence for unimodal functions. The degree of freedom of the trial vectors can be increased by controlling the scaling factors F and $|\sigma|$ with a wavelet function. We could have high degree of exploration in the early stage of evolution. Moreover, taking advantage of the fine-tuning ability of the wavelet operations, the population can have more exploitation at the late stage of searching to catch the global minimum. In short, the proposed DWM-DE is the best method to tackle unimodal functions among the DE methods covered in Table 3.4.

2. Multimodal functions with a few local minima

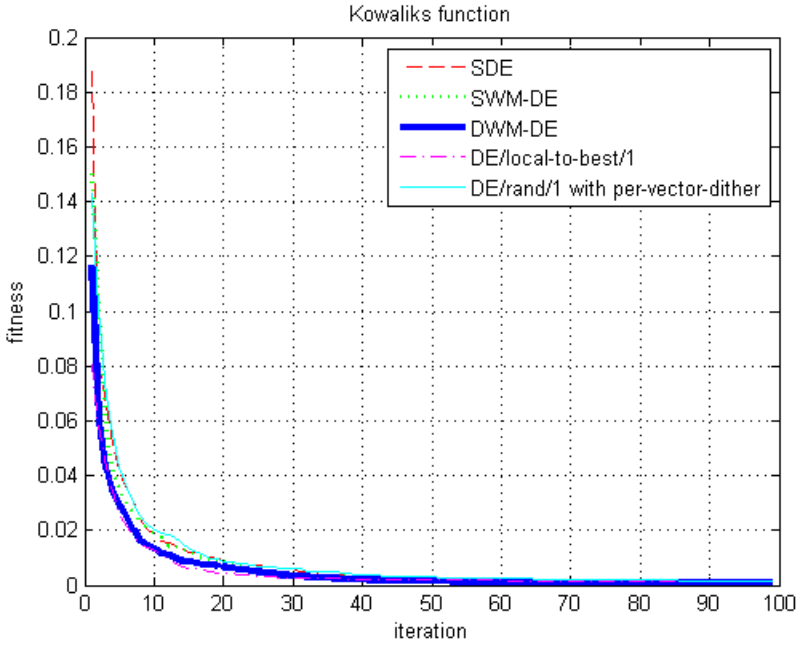
Eight multimodal functions with a few local minima are used to evaluate the five algorithms. These functions are Shekel's foxholes function, Kowalik's function, Maxican hat function, Six-hump camel back function, Hartman's family 1, Hartman's family 2, Egg holder function and Styblinski-Tang function. All of them contain some local minima within the searching space. The experimental results for these functions are listed in Table 3.5 and shown in Fig. 3.7.

Function f_{15} is the Egg-holder function, which is a well-established benchmark test function for evaluating the performance of global optimisation algorithms. It is a difficult test function for optimisation algorithms, especially when the dimension of the equation is high ($\geq$20). It contains a number of local optima. The algorithm could be trapped in some local optimal point easily. In the experiments, we set the Egg-holder function with the dimension of 20. The experimental results show that DWM-DE out-performs significantly the other four algorithms in terms of convergence speed, solution quality and solution reliability. It should be noted that when the dimension of the Egg-holder function is 2, DWM-DE just performs nearly the same as the other methods. This test function illustrates that when the dimension of the problem is high, the effect of the double wavelet mutations is more significant.

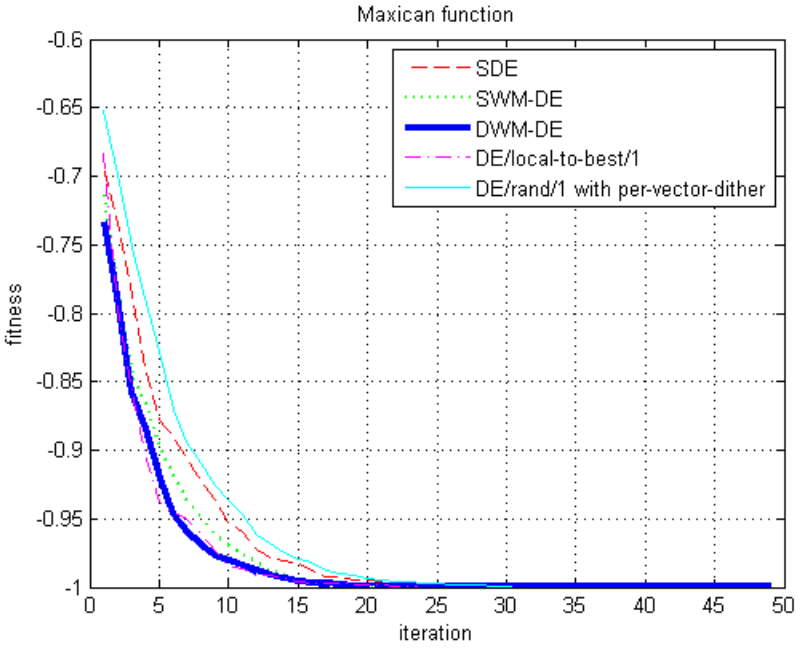
Function f_{16} is the Styblinski-Tang function. It is a 2-dimension benchmark test function with the global optimum at $f(-2.903534, -2.903534) = -78.332$. This function has a bowl shape surface with four separated regions in which each region contains one local optimum. Moreover, one of the four separated regions contains the global optimum. Because of the four separated regions, the optimiser could not recover easily when the algorithm is trapped in one of the regions in the search domain.


Shekels Foxholes function
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
50
100
150
200
250
300
350
400
450
500
0
5
10
15
20
25
30
35
40
45
50

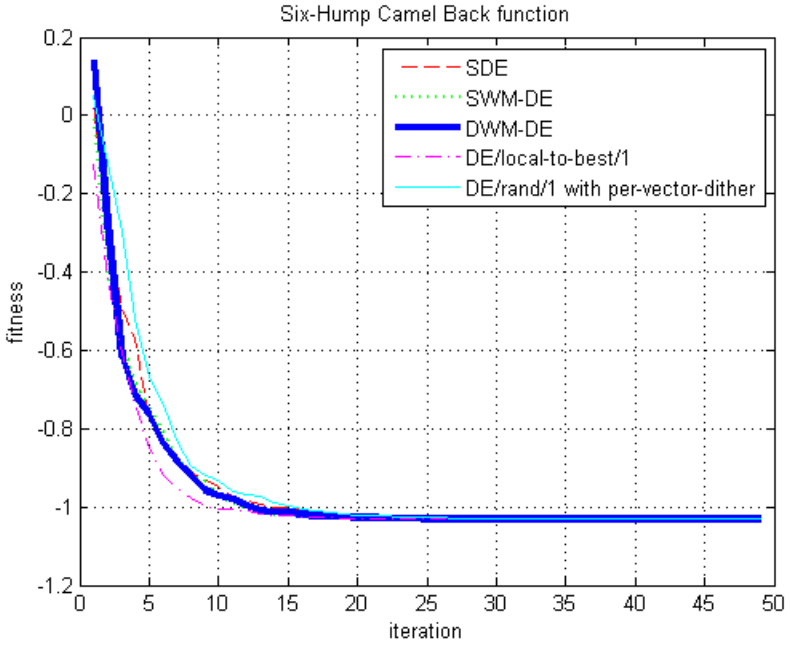
(a)


Kowaliks function
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0
10
20
30
40
50
60
70
80
90
100

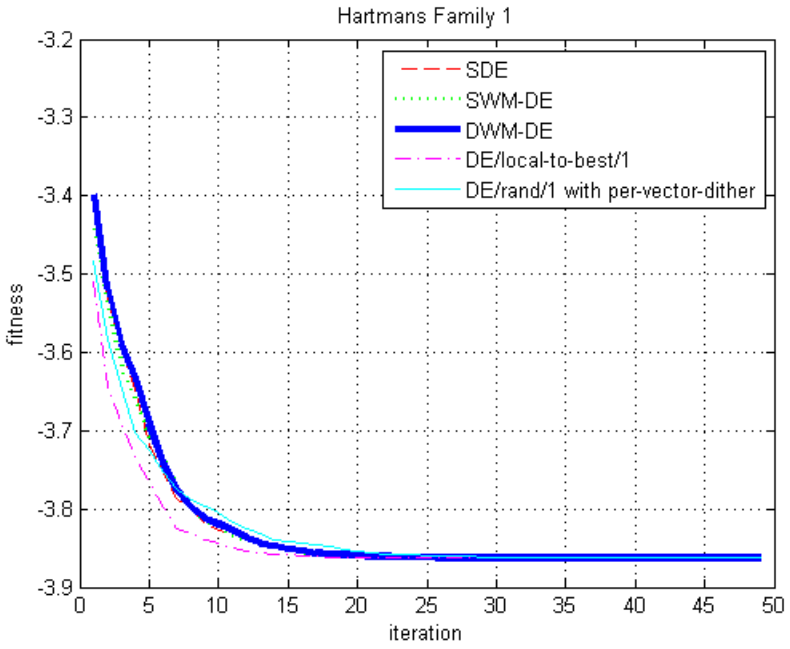
(b)


Maxican function
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
-0.6
-0.65
-0.7
-0.75
-0.8
-0.85
-0.9
-0.95
-1
0
5
10
15
20
25
30
35
40
45
50

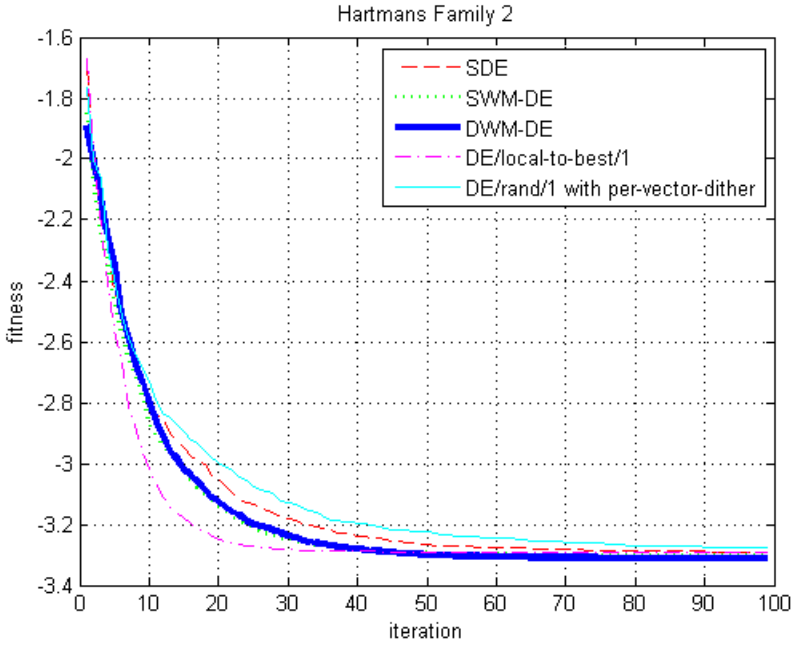
(c)


Six-Hump Camel Back function
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
5
10
15
20
25
30
35
40
45
50

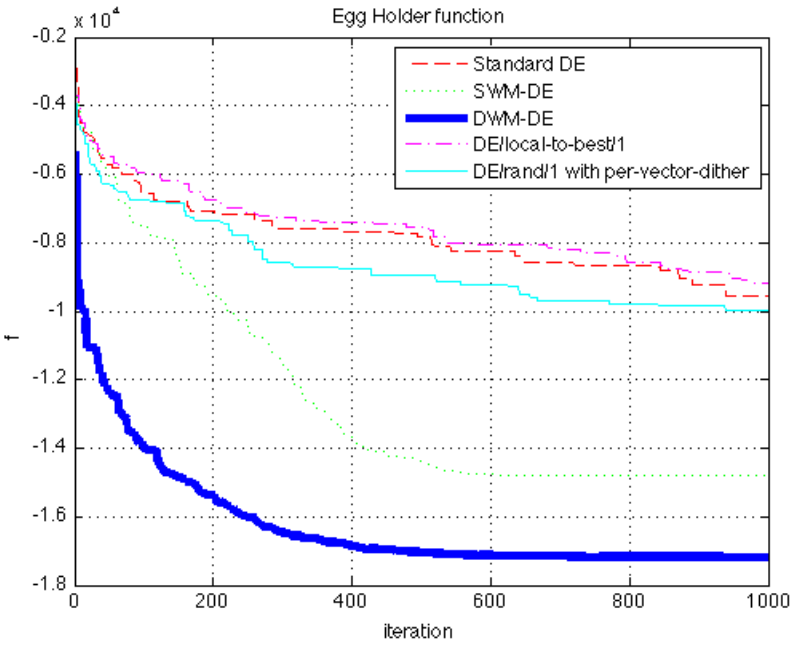
(d)


Hartmans Family 1
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
-3.2
-3.3
-3.4
-3.5
-3.6
-3.7
-3.8
-3.9
0
5
10
15
20
25
30
35
40
45
50

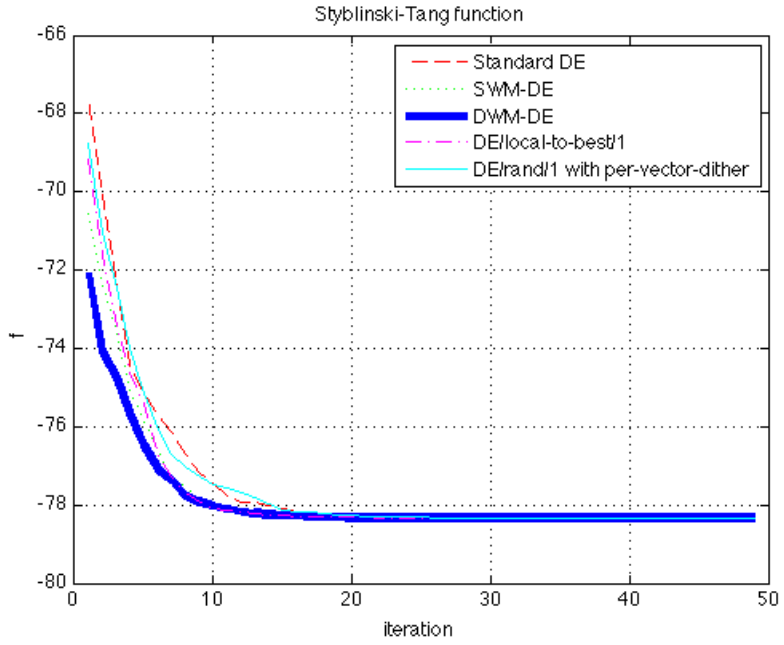
(e)


Hartmans Family 2
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
-1.6
-1.8
-2
-2.2
-2.4
-2.6
-2.8
-3
-3.2
-3.4
0
10
20
30
40
50
60
70
80
90
100

(f)



(g)



(h)

Fig. 3.7. Multimodal functions with a few local minima.

Table 3.5. Comparison between Different DE Methods for Benchmark Test Functions (Category 2).

		DWM-DE	SWM-DE	SDE	DE/ local-to-best/ 1	DE/rand/1 with per-vector -dither
f_9	Mean	0.998	0.998	0.998	0.998	0.998
	Best	0.998	0.998	0.998	0.998	0.998
	Std Dev	0	0	0	0	0
f_{10}	Mean	0.0009	0.0012	0.0011	0.0015	0.0016
	Best	0.0005	0.0004	0.0007	0.0003	0.0007
	Std Dev	0.0003	0.0011	0.0008	0.0039	0.0018
f_{11}	Mean	-1	-1	-1	-1	-1
	Best	-1	-1	-1	-1	-1
	Std Dev	0	0	0	0	0
f_{12}	Mean	-1.0316	-1.0316	-1.0316	-1.0316	-1.0315
	Best	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	Std Dev	0	0	0.0001	0	0.0002
f_{13}	Mean	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628
	Best	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628
	Std Dev	0	0	0	0	0
f_{14}	Mean	-3.3124	-3.3036	-3.2909	-3.2911	-3.2777
	Best	-3.322	-3.322	-3.322	-3.322	-3.322
	Std Dev	0.0303	0.0408	0.0475	0.0527	0.0448
f_{15}	Mean	-17185.636825	-14785.571548	-9540.044930	-9166.199635	-9981.148852
	Best	-17369.449901	-15957.227385	-11146.971689	-11490.211886	-10646.983475
	Std Dev	134.842749	775.309032	998.757469	1779.006804	513.702581
f_{16}	Mean	-78.332331	-78.332331	-78.33233	-78.33233	-78.332318
	Best	-78.332331	-78.332331	-78.332331	-78.332331	-78.332331
	Std Dev	0.000001	0.000001	0.000003	0.000003	0.000018

For all the functions of multimodal functions with a few local minima, the five algorithms offer similar performance on searching the optimal point except the Egg Holder function. DWM-DE can provide much better solution quality and convergence rate. No algorithm is trapped in the local minima. No significant enhancement is brought by the double wavelet mutations for multimodal functions. Yet, DWM-DE can still offer good performance in term of the solution quality and reliability. It is applicable for multimodal functions with a few local minima.

3. Multimodal functions with many local minima

Functions f_{17} - f_{24} are multimodal functions with many local minima. The experimental results for these functions are listed in Table 3.6 and shown in Fig. 3.8. Functions $f_{17,}$ $f_{18,}$ f_{19} are the Generalised penalised function, Generalised Rastrigin's function and Generalised Griewank's function respectively. They are widely used as test functions for global optimisation algorithms. Those functions have many local minima distributed regularly. When the function dimension increased, the number of local minima increases exponentially. In the experiments, the function contains plenty of local minima when the dimension is 30. From Fig. 3.8a-c, we can see that the rate of convergence is improved significantly when the wavelet mutation is introduced. By adding the double wavelet mutations to DE, the chance of trapping in some local minima is reduced. Moreover, by introducing the second wavelet mutation, the searching process of DWM-DE is capable of moving closely to the global minimum in the early iteration stage. The balance between the exploration and exploitation can be achieved thanks to the wavelet function's properties. The wavelet function provides a more efficient searching process with better reliability in searching the global solutions of different problems.

Function f_{20} is the Generalised Ackley's function. It modulates a cosine wave into an exponential function. It is a continuous multimodal function with a flatland and a central minimum. The result is shown in Fig. 3.8d. It shows that if the double wavelet mutations are used, the value of the fitness function drops rapidly. After 250 times of iteration, the fitness value becomes near the global minimum. It should be noticed that even DE with single wavelet mutation could not satisfactorily reach the global minimum. It shows the advantage of the double wavelet mutations on reducing the effort for the population to investigate those local minima that are far away from the global minimum.

Function f_{21} is the Schwefel's function. Most of the optimisation algorithms fail to handle this function because the global minimum is isolated from the similar best local minimum geometrically. The population is potentially prone to converge to the local minimum. The result is shown in Fig. 3.8e. Similar to functions f_{19} and f_{20}, if the double wavelet mutations are used, the convergence rate is much improved. With DWM-DE, the fitness value moves closely to the global minimum at the early iteration stage.

Function f_{22} is the Schaffer function. It is a 2-dimension benchmark function that has a single global minimum at the point $f_{22}(0,0) = 0$. It contains a large number of local minima. Searching for the global minimum is a difficult task because the value at the best local minimum and the global minimum differs by about 0.0001 only. In the experiment, although the convergence rate is not significantly improved, DWM-DE could offer much better solution quality and solution reliability.

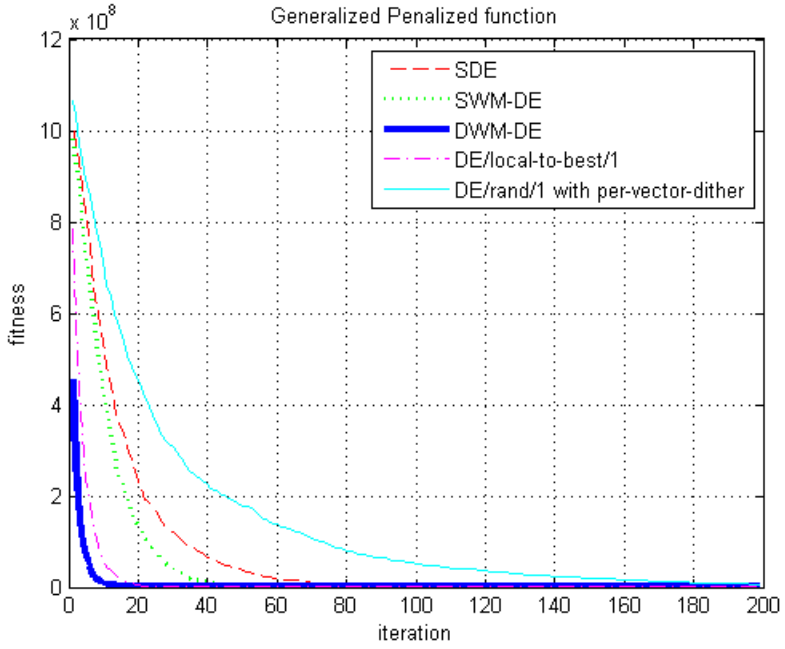
Function f_{23} is the Chichinadze function which is a 2-dimension benchmark test function with a global optimum at f (5.90133, 0.5) = −43.3159. As this function is not complex, DWM-DE just performs nearly the same as the other algorithms.

Function f_{24} is the Sine envelope sine wave function. It is also called the Schaffer function. The Sine envelope sine wave function is a multi-dimensional version of the Schaffer function. It contains a large number of local minima. Searching for the global minimum is a difficult task because the difference between the value at the best local minimum and the global minimum is about 0.001 only. The number of local optima is not well defined, and they are continuously spread around the global optimum. Theoretically, there are infinite local minima that form a number of grooves around the global minimum. In the experiments, we set the Sine envelope sine wave function's dimension to 20. The experimental results show that DWM-DE significantly out-performs the other algorithms in terms of convergence speed, solution quality and solution reliability. It should be noted that

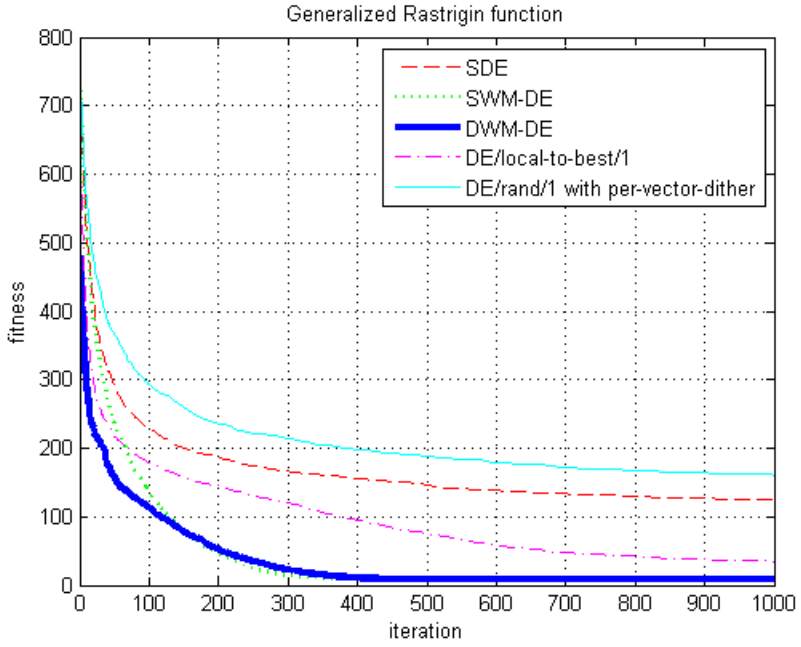
if the dimension of the Sine envelope sine wave function is two (which is the Schaffer function f_{22}), the DWM-DE just performs nearly the same as the other algorithms in term of the convergence rate. It illustrates that for the high dimensional problem, the performance enhancement of the double wavelet mutations is more significant.

Table 3.6. Comparison between Different DE Methods for Benchmark Test Functions (Category 3).

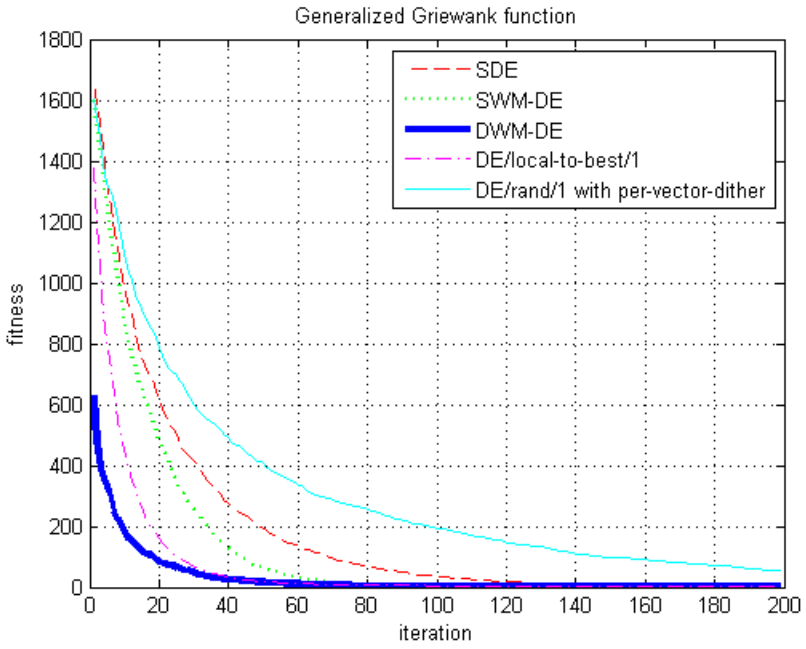
		DWM-DE	SWM-DE	SDE	DE/ local-to-best/ 1	DE/rand/1 with per-vector -dither
f_{17}	Mean	0.4883	67682.17	207.3875	147.6601	6835946
	Best	0.1582	8.0881	22.4585	4.7027	149.9788
	Std Dev	0.2989	289798.5	291.9504	391.5459	450.1213
f_{18}	Mean	8.3213	9.6949	124.7772	34.9675	160.9108
	Best	0.0063	3.3497	87.5478	16.9349	136.2236
	Std Dev	4.4707	3.155	10.5849	11.9889	10.8145
f_{19}	Mean	1.0243	2.0471	2. 83	3.0293	54.1213
	Best	0.8751	1.1207	1.5661	1.0394	33.3442
	Std Dev	0.0576	1.1589	0.4224	1.9004	10.991
f_{20}	Mean	0.3199	3.983	0.7723	4.6762	17.728
	Best	0.0271	0.2236	0.0261	2.2245	7.1453
	Std Dev	0.3194	3.5579	2.9323	1.5461	3.687
f_{21}	Mean	-12568.8	-12254.1	-12475.8	-10328.1	-10128.1
	Best	-12569.5	-12563.5	-12569	-11004.5	-11272.4
	Std Dev	0.6883	181.3254	145.1123	391.6365	560.2403
f_{22}	Mean	0.000001	0.00583	0.003886	0.002062	0.001943
	Best	0	0	0	0	0
	Std Dev	0.000001	0.005322	0.005322	0.004286	0.004345
f_{23}	Mean	-43.315756	-43.31493	-43.315719	-43.315739	-43.310539
	Best	-43.315859	-43.315862	-43.315866	-43.315859	-43.315487
	Std Dev	0.000219	0.002076	0.000187	0.000221	0.005238
f_{24}	Mean	1.629914	1.855742	9.812808	8.221035	10.605929
	Best	0.644249	0.75169	8.753928	6.868805	9.863131
	Std Dev	0.572477	0.624029	0.342526	0.589078	0.250125


Generalized Penalized function
x 10^8
fitness
iteration
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

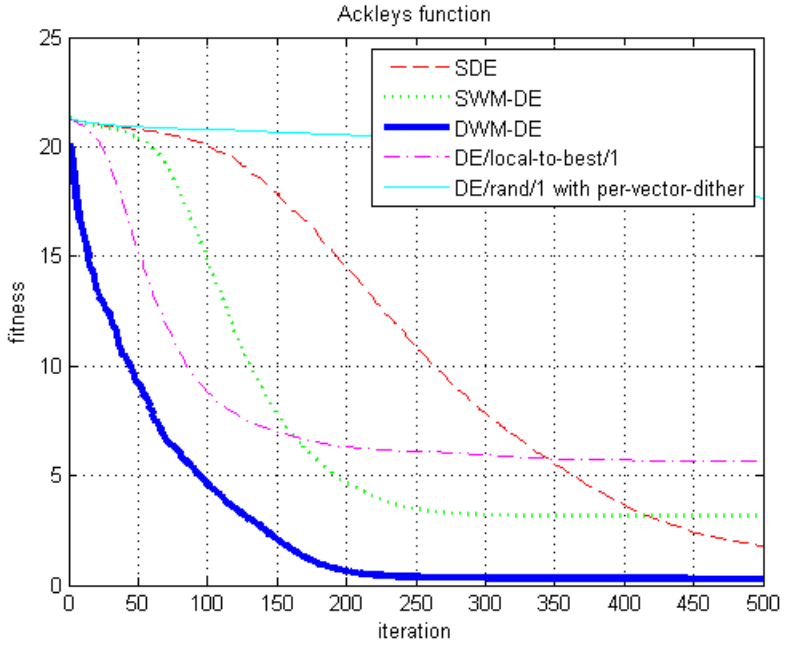
(a)


Generalized Rastrigin function
fitness
iteration
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

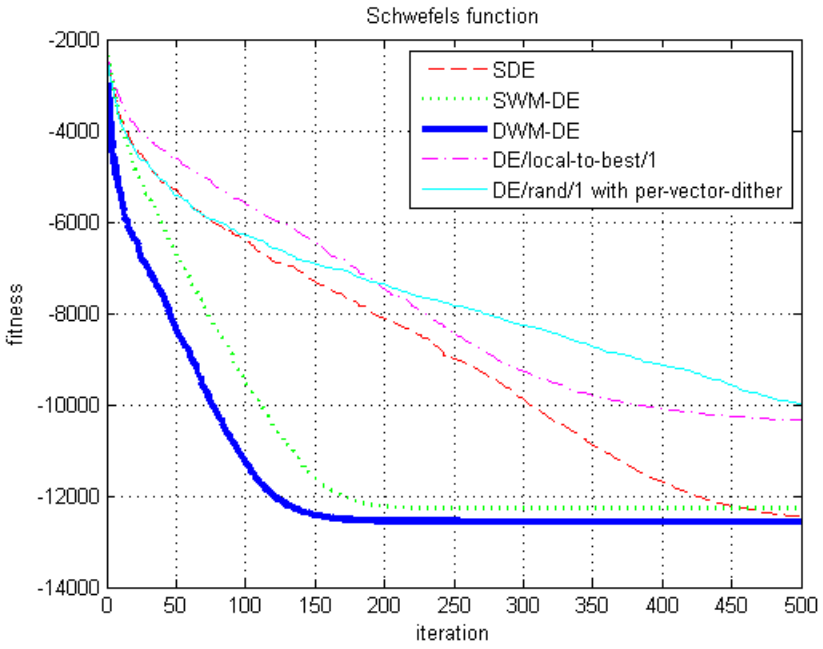
(b)


Generalized Griewank function
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
200
400
600
800
1000
1200
1400
1600
1800
0
20
40
60
80
100
120
140
160
180
200

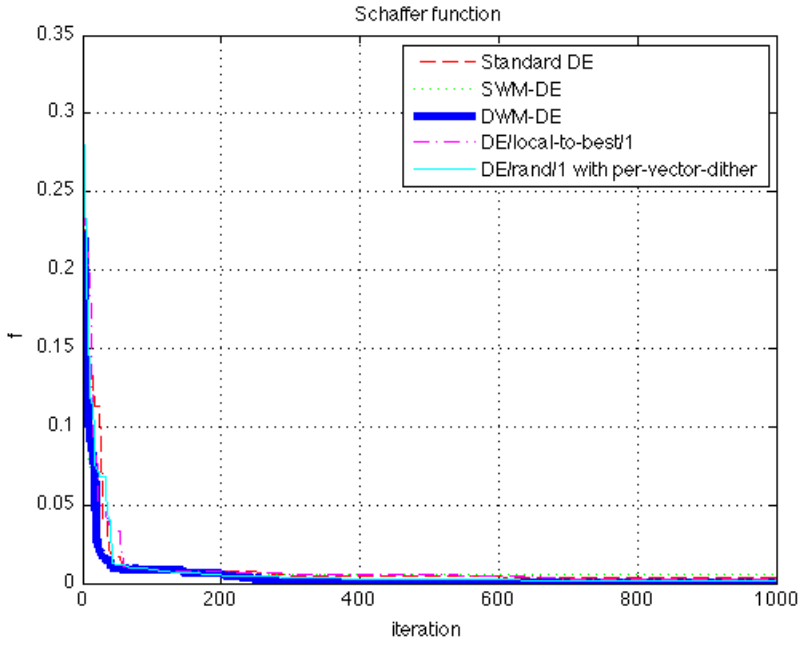
(c)


Ackleys function
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
5
10
15
20
25
0
50
100
150
200
250
300
350
400
450
500

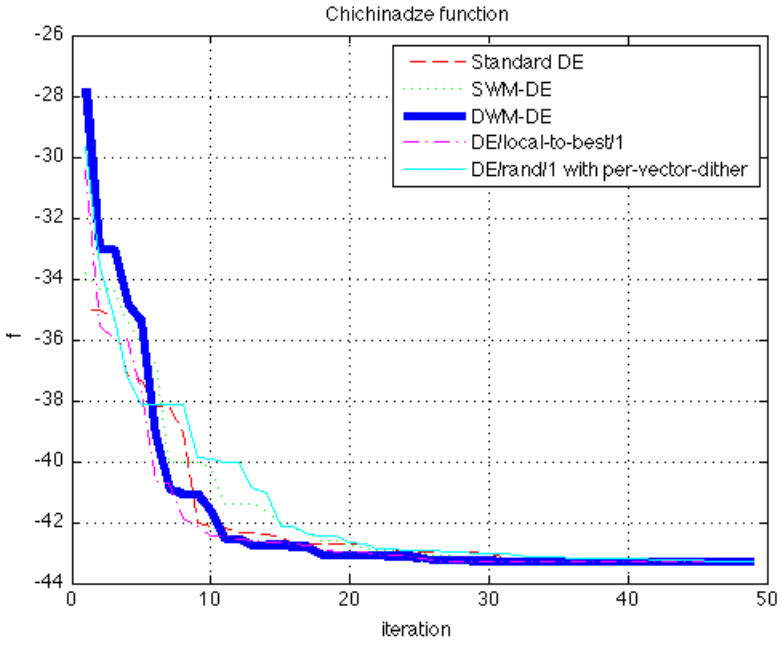
(d)


Schwefels function
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
-2000
-4000
-6000
-8000
-10000
-12000
-14000
0
50
100
150
200
250
300
350
400
450
500

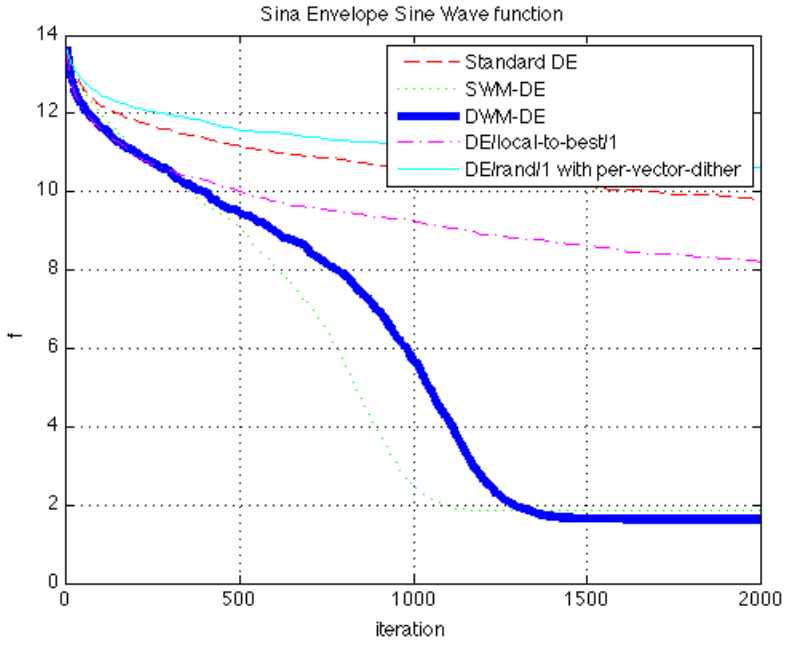
(e)


Schaffer function
Standard DE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
f
iteration
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
200
400
600
800
1000

(f)



(g)



(h)

Fig. 3.8. Multimodal functions with many local minima.

To summarise, the performance enhancement of the proposed DWM-DE for multimodal functions with many local minima is significant. In particular, DWM-DE can offer much better searching operation to avoid the chance of get trapped in local optimum.

4. Functions with shift and rotate

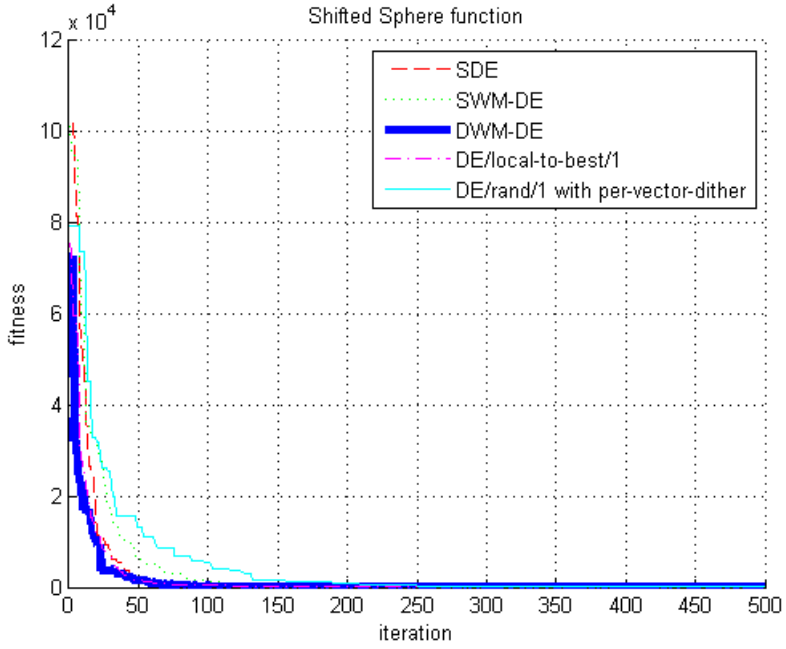
Five functions with shift and rotate is introduced to test the performance of the DWM-DE. Three functions are unimodal functions. They are the Shifted sphere model, Shifted Schwefel's problem 1.2, and Shifted rotated high conditioned elliptic function. Two functions are multimodal functions. They are the Shifted Rosenbrock's function, and Shifted Rastrigin's function.

Some of the benchmark functions in the previous three categories have drawbacks as the elements in the optimum vector might have the same value for different dimensions owing to the symmetry nature. The global optimum is normally located at the centre of the searching domain. Some optimisers are designed to converge to the centre of the searching domain even no searching direction is provided. Hence, this kind of benchmark functions might not be good to evaluate the real performance of optimisers. To overcome these drawbacks, shift and rotate are introduced to the benchmark test functions to generate optimum points with different numerical values. The optimum point is not lying at the centre of the searching domain.

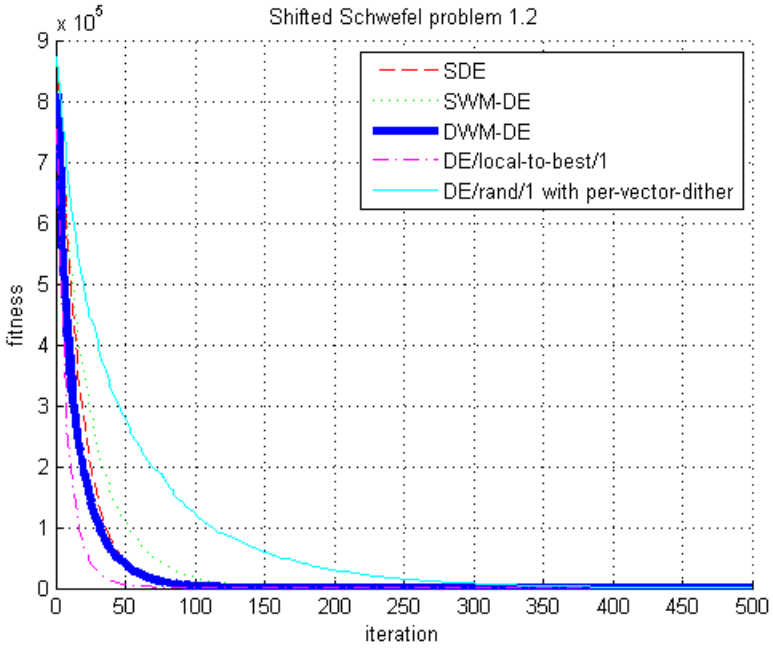
The experimental results for these functions are listed in Table 3.7 and shown in Fig. 3.9. The results show that the proposed DWM-DE is still able to offer the best performance. The double wavelet mutations offer significant improvement on searching the optimum point. The solution reliability and quality offered by DWM-DE are good.

Table 3.7. Comparison between Different DE Methods for Benchmark Test Functions (Category 4).

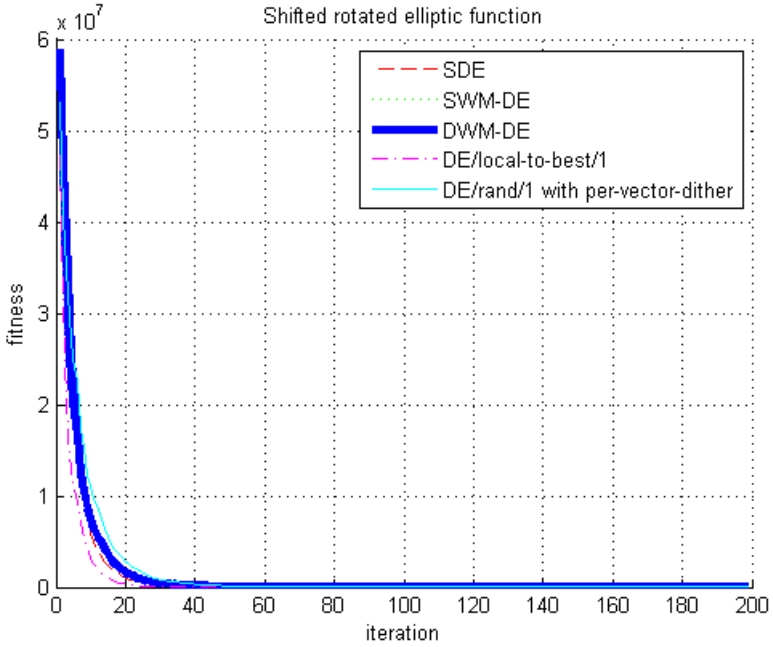
		DWM-DE	SWM-DE	SDE	DE/ local-to-best/ 1	DE/rand/1 with per-vector -dither
f_{25}	Mean	0.000742	0.10379	41.30417	349.7417	26.59024
	Best	0.000244	0.008027	0.019934	7.378115	13.61648
	Std Dev	0.000302	0.074309	135.404	389.9865	10.21125
f_{26}	Mean	0.011645	3.756602	199.2697	374.9674	575.92
	Best	0.002418	0.213621	0.123243	3.066111	259.5999
	Std Dev	0.004996	3.114776	466.9468	429.4013	163.9056
f_{27}	Mean	0.00149	39.90172	194.3431	0.004001	2.516334
	Best	0.000056	0.082571	0.000341	0.000096	0.396908
	Std Dev	0.001327	131.0532	815.1713	0.028289	1.447321
f_{28}	Mean	4.760516	7.149275	5.435293	4.099365	11.69429
	Best	0.001795	0.000027	0.009373	0.00047	0.008133
	Std Dev	8.102966	10.73707	8.352652	6.416724	15.96312
f_{29}	Mean	8.267052	126.6133	10.32804	35.18819	159.3997
	Best	2.004883	103.3226	5.175221	12.95799	127.034
	Std Dev	2.98555	8.730934	3.015084	10.97512	10.40491



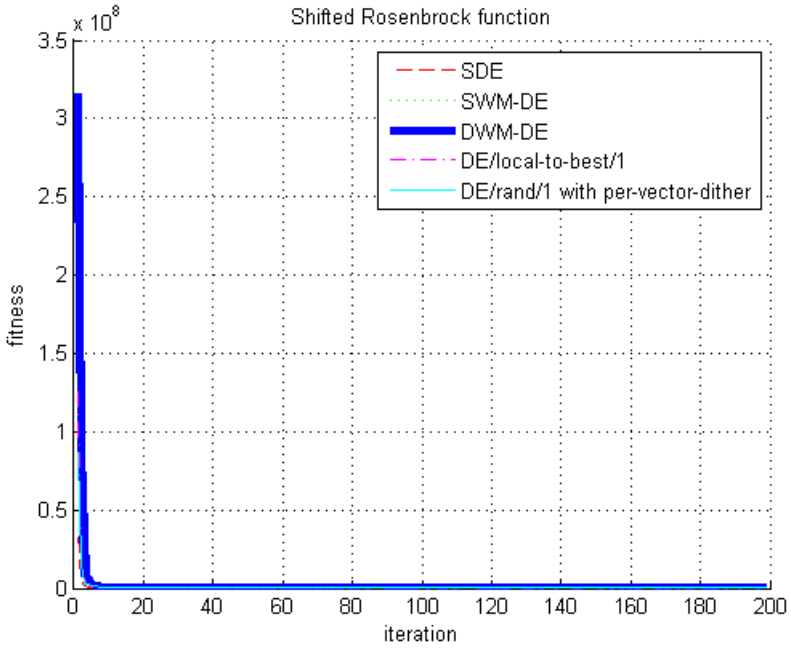
(a)


Shifted Schwefel problem 1.2
x 10^5
fitness
iteration
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

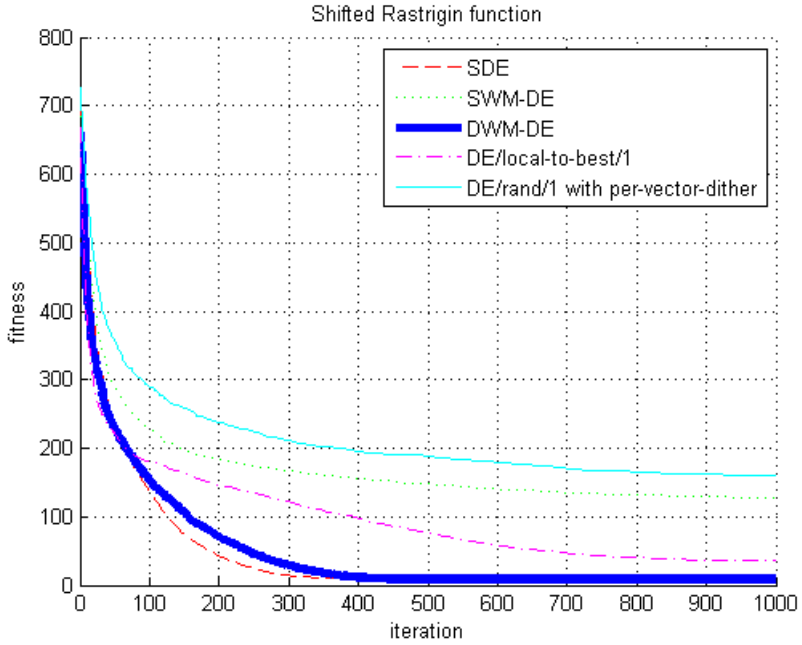
(b)


Shifted rotated elliptic function
x 10^7
fitness
iteration
SDE
SWM-DE
DWM-DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

(c)



(d)



(e)

Fig. 3.9. Functions with shift and rotate.

D. The T-Test

Student T-Test determines whether two groups of data are statistically different from each other. It is particularly useful when the analysis involves two groups of random data that follow a normal distribution. The *t*-value generated by the Student T-Test is a ratio between the means difference and standard deviations of the two sample groups:

$$t = \frac{\overline{\alpha_2} - \overline{\alpha_1}}{\sqrt{\left(\frac{\sigma_2^2}{\xi_{degree}}\right) + \left(\frac{\sigma_1^2}{\xi_{degree}}\right)}} \tag{3.17}$$

where $\overline{\alpha_{g1}}$ and $\overline{\alpha_{g2}}$ are respectively the mean of the best fitness values of the two methods; σ_1 and σ_2 are respectively are the standard deviations of the best fitness values of the two methods respectively; and ξ_{degree} is the number of samples in the group. For 50 trials, we have the ξ_{degree} equal to 50. Table 3.8 shows the *t*-values of DWM-DE with the other methods. If the *t*-value is undefined, it is denoted as N/A. If the value of *t* is equal to or larger than 1.645, we have 95% confidence that a significant difference between the two algorithms is observed. In most of the cases, the value of *t* in the experiment is larger than 1.645. We can conclude that there are significantly differences of the DWM-DE with other methods.

Table 3.8. *t*-Value between DWM-DE and Other Algorithms.

Functions	*t*-value between DWM-DE and SWM-DE	*t*-value between DWM-DE and SDE	*t*-value between DWM-DE and DE/ local-to-best/ 1	*t*-value between DWM-DE and DE/rand/1 with per-vector -dither
f_1	3.106535	4.396617	7.568123	29.28583
f_2	11.97319	274.8664	18.24283	54.5784
f_3	5.539252	41.57059	9.705914	44.80157
f_4	5.90861	19.61271	10.67363	22.19623
f_5	24.0406	9.866993	33.41888	72.54976
f_6	3.489848	36.80786	7.659204	58.28451
f_7	1.808415	2.034921	5.024374	8.507825
f_8	5.49971940	7.378506	N/A	7.172083
f_9	N/A	N/A	N/A	N/A
f_{10}	1.860521	1.655212	1.084652	2.712445
f_{11}	N/A	N/A	N/A	N/A
f_{12}	N/A	0	N/A	3.535534
f_{13}	N/A	N/A	N/A	N/A
f_{14}	1.224414	2.69834	2.477622	4.536719
f_{15}	21.5656321	53.64307	31.78391	95.91962
f_{16}	0	2.236068	2.236068	5.09902
f_{17}	1.651429	5.011117	2.657828	10.73876
f_{18}	4.521753	85.52812	39.58508	109.6375
f_{19}	6.232955	20.30214	7.456842	34.15952
f_{20}	7.250986	1.084521	19.51148	33.26134
f_{21}	11.0344	8.376944	38.8019	34.21467
f_{22}	7.74469251	5.1618	3.40025	3.160417
f_{23}	2.79791516	0.908512	0.38636	7.036571
f_{24}	1.88564606	86.73324	56.73815	101.5953
f_{25}	1866.1629	0.2252802	0.229957	25.50071
f_{26}	38.6004	0.09139	0.20335	2.14371
f_{27}	0.2323165	0.0292461	313.0812	120.0552
f_{28}	1.3201740	0.4982667	-0.61887	2.163564
f_{29}	138.99749	11.447236	20.80993	128.9796

E. Sensitivity of the Shape Parameter for DWM-DE

For different values of the shape parameter ζ_{wm}, the proposed DWM-DE may perform differently. In this section, we are going to evaluate the effect to the performance for different values of the shape parameter ζ_{wm}. Table 3.9 lists the mean values of the solution obtained by DWM-DE under different setting of the shape parameter ζ_{wm}. In these experiments, the parameter λ is fixed at 10000. Five values of ζ_{wm} are used to test the performance: 0.2, 0.5, 1, 2, and 5. Referring to the experimental result reported in Table 3.9, the value of ζ_{wm} does not affect the performance significantly in some functions. However, the performance of the proposed DWM-DE is sensitive to the value of the parameter ζ_{wm} in some other functions. We are not able to generate any rules or methods to choose the best value of the parameter ζ_{wm}. As a result, we suggest the user to set the value of the parameter ζ_{wm} to 1 to simplify the usage of DWM-DE.

Table 3.9. Sensitivity of Shape Parameter for Wavelet Mutation.

Functions	ζ_{wm} =0.2	ζ_{wm} =0.5	ζ_{wm} =1	ζ_{wm} =2	ζ_{wm} =5
f_1	0.1022	0.4705	0.5902	1.4617	0.6216
f_2	0.0065	0.0144	0.0961	0.3188	16.7615
f_3	0	0	0	0.12	1.76
f_4	0.0329	0.033	0.0385	0.044	0.0582
f_5	0.4248	0.7679	1.4127	2.771	4.5791
f_6	0.3018	0.5405	0.388	0.937	0.7229
f_7	-1	-1	-1	-0.9999	-0.9798
f_8	0.998	0.998	0.998	0.998	7.2744
f_9	-1.913223	-1.913223	-1.913223	-1.913223	-0.564333
f_{10}	0.0014	0.001	0.0009	0.0013	0.0016
f_{11}	-1	-1	-1	-1	-1
f_{12}	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
f_{13}	-3.8628	-3.8627	-3.8628	-3.8623	-3.862
f_{14}	-3.3121	-3.312	-3.3124	-3.3186	-3.3059

f_{15}	-17369.000322	-17369.449901	-17369.449901	-17369.449901	-15666.236432
f_{16}	-78.332331	-78.332331	-78.332331	-78.332331	-78.332331
f_{17}	0.1728	0.4728	0.4883	1.308	7.9534
f_{18}	5.4248	4.9128	8.3213	10.4683	10.9009
f_{19}	0.9953	1.0694	1.0243	0.9733	0.9356
f_{20}	0.0457	0.2032	0.3199	1.2207	1.7944
f_{21}	-12569.4	-12569.1	-12568.8	-12567.5	-12554.9
f_{22}	-3.322	-3.322	-3.322	-3.322	-3.322
f_{23}	-17369.449901	-17369.449901	-17369.449901	-17369.449901	-17369.449901
f_{24}	-78.332331	-78.332331	-78.332331	-78.332331	-78.332331
f_{25}	1.334	0.975	0.000742	0.00642	2.03452
f_{26}	3.43453	2.45446	0.011645	0.34354	2.33325
f_{27}	6.56546	3.46554	0.00149	4.35534	5.466645
f_{28}	8.76945	6.742316	4.760516	4.44436	4.770212
f_{29}	10.43546	9.445051	8.267052	11.234099	15.243094

F. Sensitivity of the Parameter λ for DWM-DE

For different values of the parameter λ, the proposed DWM-DE may perform differently. In this section, we are going to evaluate the effect to the performance for different values of the parameter λ. Table 3.10 lists the mean values of the solution obtained by DWM-DE under different setting of the parameter λ. In these experiments, the value of ζ_{wm} is fixed at 1. Four values of the parameter λ are used to test the performance: 100, 1000, 10 000, and 100 000. Referring to the experimental result reported in Table 3.10, the value λ does not affect the performance significantly in most of the functions. Similar to the parameter ζ_{wm}. We are not able to generate any rules or methods to choose the best value of the parameter λ. As a result, we suggest the user to set the value of the parameter λ to 10000 to simplify the usage of the DWM-DE.

Table 3.10. Sensitivity of Parameter λ for Wavelet Mutation.

Functions	λ =100	λ =1000	λ =10000	λ =100000
f_1	0.3202	0.8152	0.5902	1.4945
f_2	0.0131	0.0357	0.0961	0.0359
f_3	0	0.02	0	0.02
f_4	0.0405	0.0395	0.0385	0.0395
f_5	0.6381	1.0508	1.4127	1.5449
f_6	0.4413	0.7524	0.388	0.8447
f_7	-1	-0.9997	-1	-1
f_8	-1.913223	-1.913223	-1.913223	-1.913223
f_9	0.998	0.998	0.998	0.998
f_{10}	0.0011	0.001	0.0009	0.0011
f_{11}	-1	-1	-1	-1
f_{12}	-1.0316	-1.0316	-1.0316	-1.0316
f_{13}	-3.8628	-3.8619	-3.8628	-3.8623
f_{14}	-3.3122	-3.3145	-3.3124	-3.3211
f_{15}	-16369.1234	-17369.449901	-17369.449901	-17369.449901
f_{16}	-78.332331	-78.332331	-78.332331	-78.332331
f_{17}	0.3142	0.6711	0.4883	0.9591
f_{18}	6.7112	7.7008	8.3213	8.3281
f_{19}	1.0369	1.0698	1.0243	1.0726
f_{20}	0.2116	0.3268	0.3199	0.4361
f_{21}	-12569.1056	-12568.6379	-12568.8411	-12568.2966
f_{22}	-3.322	-3.322	-3.322	-3.322
f_{23}	-17369.449901	-17369.449901	-17369.449901	-17369.449901
f_{24}	-78.332331	-78.332331	-78.332331	-78.332331
f_{25}	3.97655	0.000742	0.000944	0.05642
f_{26}	3.56753	0.011645	0.013333	0.343434
f_{27}	7.22233	0.00149	0.02345	4.35732
f_{28}	7.70043	4.760516	4.760888	4.44367
f_{29}	15.11532	8.267052	8.267052	12.20008

IV CONCLUSION

The design detail of the DWM-DE algorithm is discussed in this chapter. To help the evolution, two stages of wavelet operation are embedded in the standard DE algorithm. By introducing the double wavelet mutations in DE, the searching process is enhanced by offering an effective balance between the exploration and exploitation of the searching space for better solution reliability and quality. In the DE mutation operation, a wavelet function is employed to control the mutation factor F. In the DE crossover operation, a wavelet-based second mutation mechanism is proposed to modify the trial vectors within the population. A suite of 29 benchmark test functions is employed to test the performance of the proposed DWM-DE. Experiment results show that the proposed DWM-DE is a useful tool for solving optimisation problems, and it offers better results in terms of solution reliability, solution quality and convergence rate. The experiment results reflect that DWM-DE is particularly suitable for complex problems with a high dimension ($\geq$20).

Chapter 4

INTELLIGENT OPTIMISER WITH WAVELET-MUTATED DIFFERENTIAL EVOLUTION

I INTRODUCTION

In this chapter, we discuss a proposed intelligent optimiser. It operates with two identical optimisation algorithms and a fuzzy controller. The two optimisation algorithms are operated in parallel with different initial conditions. The fuzzy controller is used to control the parameters of the two optimisation algorithms. The parallel implementation framework aims at enhancing the optimisation performance. The optimisation algorithm is enhanced from the standard Differential Evolution (DE) algorithm by introducing a wavelet mutation in the DE crossover operation. We name this algorithm as Wavelet-Mutated Differential Evolution (WM-DE). The wavelet mutation operation aims to achieve a balance between the exploration and exploitation of the searching space. In the early stage of searching, we want more exploration while more

exploitation is desired at the later stage of searching. Exploration brings a wider searching in a larger solution space. Exploitation brings the searching of the solution space in a small area to reach a fine-tuned solution. This mechanism can be realised by a wavelet function, which is a mathematical tool to model seismic signals in a finite domain. The performance of DE can be improved by the wavelets properties in terms of convergence speed, solution quality and solution reliability in a statistical sense. A detailed discussion on wavelet-based mutation has been given in Chapter 3.

In Chapter 3, we have proposed a novel optimisation algorithm called DWM-DE, which can successfully enhance the DE performance on solution reliability, convergence speed and solution quality. However, the performance of DWM-DE still depends quite much on some control parameters and the setting of the initial conditions. To overcome this drawback, a fuzzy controller is employed in the proposed intelligent optimiser to control the parameter values adaptively during the progress of searching. To reduce the dependence on initial conditions of the optimisation algorithm, a parallel implementation framework involving two WM-DE engines is proposed. To reduce the number of control parameters as a compensation for the increase of complexity brought by the fuzzy controller, we propose to use a single wavelet operation instead of double. The resulting system consists of two WM-DE engines running in parallel at different initial conditions to tackle the same objective function. The fuzzy controller in the proposed intelligent optimiser captures the on-line population information from the two WM-DE engines. The difference between them is used to determine the parameter values of the two engines for the next generation of evolution. The Student T-Test method is introduced to analyse the population information from two WM-DE engines.

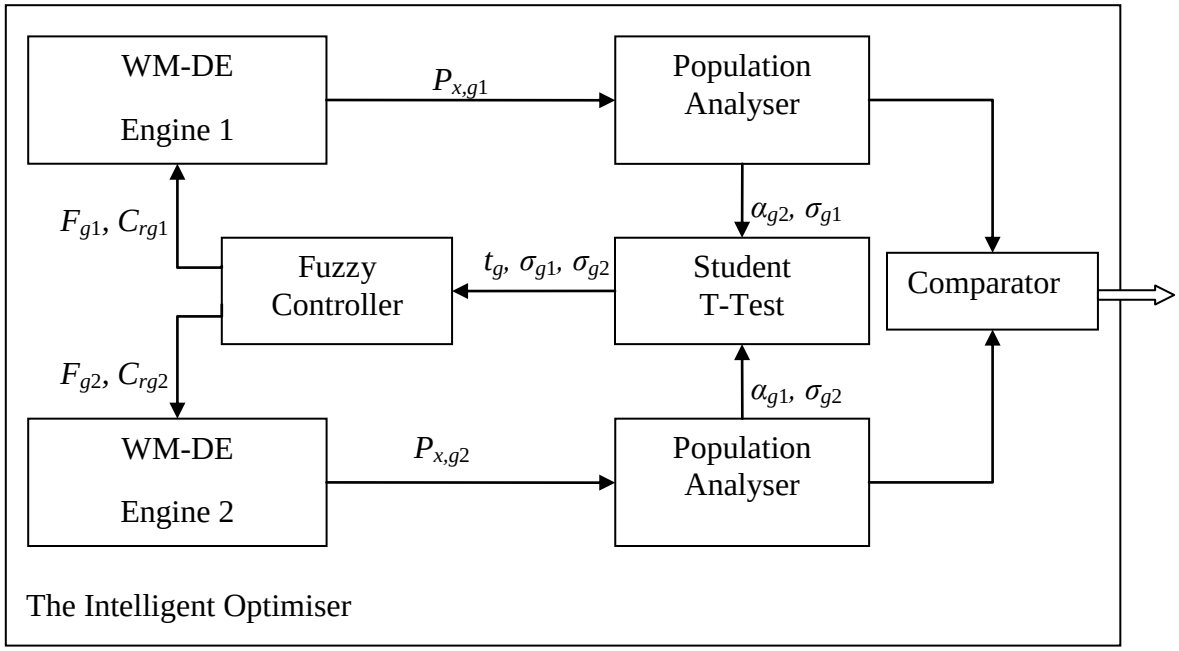
Student T-Test is a statistical method to determine whether the data of two groups are statistically different from each other in terms of their mean values. In the proposed intelligent optimiser, the Student T-Test method is used to identify the population difference in terms of mean fitness values and generate a *t*-value. Then, the individual population standard deviations and the *t*-value act as the inputs of an internal fuzzy controller to determine the next generation's F and C_r values for the two WM-DE algorithms. In practice, the two WM-DE engines act as a pairing system with additional searching information shared between each other. The result is a closed-loop adaptive control system that supports the intelligent optimiser for better performance. Thanks to the Student T-Test analysis and the fuzzy controller, the solution reliability can be enhanced when the fuzzy controller tries to minimise the *t*-value. A smaller *t*-value means a smaller difference between the populations of the two WM-DE engines, which keep guiding each other to the optimal point.

The organisation of this chapter is as follows. The proposed intelligent optimiser and the development of the fuzzy controller are presented in Section II. Section III, discuss the experimental results on applying the intelligent optimiser to 29 benchmark test functions. A conclusion is drawn in Section IV.

II INTELLIGENT OPTIMISER WITH WAVELET-MUTATED DIFFERENTIAL EVOLUTION

The proposed intelligent optimiser has an implementation framework that contains several parts. Fig. 4.1 shows the block diagram of the proposed intelligent optimiser. In addition, the pseudo code for its implementation is given in Fig. 4.2. The major parts in the intelligent

optimiser are two WM-DE engines. On doing the optimisation, the two WM-DE engines run in parallel to tackle the same objective function with two different initial populations. The initial populations are generated randomly at the beginning of the searching process. When the two WM-DE engines are operating, their individual populations at each generation (iteration) will be analysed by the population analysers as shown in Fig. 4.1.



$P_{x,g1}$: Current population from WM-DE Engine 1
$P_{x,g2}$: Current population from WM-DE Engine 2
F_{g1}: Next F for WE-DE Engine 1
F_{g2}: Next F for WE-DE Engine 2
C_{rg1}: Next C_r for WE-DE Engine 1
C_{rg2}: Next C_r for WE-DE Engine 2
α_{g1}: Mean of fitness from WE-DE Engine 1's population
α_{g2}: Mean of fitness from WE-DE Engine 2's population
σ_{g1}: Standard deviation of fitness from WE-DE Engine 1's population
σ_{g2}: Standard deviation of fitness from WE-DE Engine 2's population
t_g: Student T-test result based on populations of WE-DE engines

Fig. 4.1. Block diagram of the intelligent optimiser.

```
begin
   Initialise the population for WM-DE Engine 1
   Initialise the population for WM-DE Engine 2
   while (not termination condition) do
      begin
         WM-DE Engine 1 operation for one iteration
         Determine the values of α_g1 and σ_g1
         WM-DE Engine 2 operation for one iteration
         Determine the values of α_g2:  and σ_g2
         Calculate t_g by (4.3)
         Determinate the new value of F_g1 by the fuzzy controller
         Determinate the new value of F_g2 by the fuzzy controller
         Determinate the new value of C_rg1 by the fuzzy controller
         Determinate the new value of C_rg2 by the fuzzy controller
      end
end
```

Fig. 4.2. Pseudo code of the intelligent optimiser.

In the population analyser, the fitness value of each vector is calculated. Then, the mean and standard deviation of fitness values within the population of each WM-DE engine are found. The Student T-Test algorithm analyses the difference between the two populations and generates a *t*-value. The *t*-value and the standard deviations obtained from the two populations are input to the internal fuzzy controller, which determines the F and C_r values for the WM-DE engines for the next iteration (generation). As a result, a closed-loop adaptive intelligent optimiser is realised. At the end of the evolution, a comparator is employed to compare the results from the two WM-DE engines and adopt the best solution as the finally result of the intelligent optimiser. The detailed implementation of the intelligent optimiser is given as follows.

A. Population Analyser

The major objective of the population analyser is to estimate the movement of the population. In the proposed intelligent optimiser, each WM-DE engines move their populations within their individual searching domains. In the ideal case, after several times of iteration, all the population should converge to the area near the global optimal point. To understand the progress of searching of the WM-DE engine, we could determine the mean and standard deviation of the fitness values of the vectors in the population. Along the searching process, the standard deviation of the fitness should be gradually decreasing. In the proposed system, two WM-DE engines are operated in parallel to handle the same objective function and solution space. In the later stage of searching, the difference between the means from the two WM-DE engines should be very small. Both the mean and the standard deviation are thus used to analyse the progress of searching.

The normalised mean fitness value of the current generation within a WM-DE engine is given by:

$$\alpha_g = \frac{1}{N_p}\sum_{i}^{N_p}\left\| f(x_{i,g})\right\| \tag{4.1}$$

where N_p is the number of vectors in the current generation, $\|\cdot\|$ denotes the l_2 norm, and $x_{i,g}$ is one of the vectors in the current iteration. The normalised standard deviation of the fitness value of the current generation within a WM-DE engine is given by:

$$\sigma_g = \sqrt{\frac{1}{N_p}\sum_{i}^{N_p}\left(\left\| f\left(x_{i,g}\right)\right\| - \alpha_g\right)^2} \tag{4.2}$$

B. Student T-Test

In the proposed intelligent optimiser, the Student T-Test is employed to identify the population difference between the two WM-DE engines. The Student T-Test is a useful method to perform Hypothesis Test (HT) in the field of statistical research. HT works by collecting data and measuring the difference between the particular sets of data to prove the null hypothesis, which determines an initial guess of some experimental result. If the initial guess of the result is correct, we label the null hypothesis as TRUE, otherwise as FALSE.

Student T-Test determines whether the data of two groups are statistically different from each other in terms of their mean values. It is particularly useful when the analysis involves two groups of random data that follow a normal distribution. The *t*-value generated by the Student T-Test is a ratio between the means difference and standard deviations of the two sample groups:

$$t_g = \frac{\overline{\alpha_{g2}} - \overline{\alpha_{g1}}}{\sqrt{\left(\frac{\sigma_{g2}^2}{\xi_{degree}}\right) + \left(\frac{\sigma_{g1}^2}{\xi_{degree}}\right)}} \tag{4.3}$$

where t_g is the *t*-value of the *g*-th generation; $\overline{\alpha_{g1}}$ and $\overline{\alpha_{g2}}$ are respectively the mean values from the WM-DE engine one and WM-DE engine two; σ_{g1} and σ_{g2} are respectively the standard deviations from the WM-DE engine one and WM-DE engine two; and ξ_{degree} is the number of samples in the group.

The *t*-value can be zero, positive and negative. The *t*-value being zero means that there is no difference between the two groups. To make the *t*-value meaningful, we need to consider the significance of the HT to assure the result is statistically meaningful. To understand the

significance, we need to set an alpha level, which is generally considered as a risk level. In most of the scientific research, we set the alpha level at 0.05. The alpha level governs the probability that the means of the two tested groups have a statistically significant difference. The significance of *t*-value changes with the degree of freedom ξ_{degree}, which is the sum of the number of samples in the two tested groups minus two. Given the alpha level, the degree of freedom ξ_{degree}, and the *t*-value, we can determine whether the *t*-value is significant enough to prove the null hypothesis.

In the proposed optimiser, the value of ξ_{degree} is determined by the following equation:

$$\xi_{\mathrm{deg}ree} = N_{p1} + N_{p2} - 2 \tag{4.4}$$

where N_{p1} is the number of vectors in the population of the WM-DE engine 1 and N_{p2} is the number of vectors in the population of the WM-DE engine 2. When we are using the intelligent optimiser, we have a null hypothesis that there should not have any significant difference between the two WM-DE engines as they are solving the same objective function. Along the searching operation, the *t*-value should be kept small and less than the significant level. However, in most of the cases of solving different optimisation problems, the *t*-value could not be kept small because of the randomness in the searching process of WM-DE. Therefore, an internal fuzzy controller is employed in the proposed optimiser to control the searching process of WM-DE in order to reduce the chance that the two populations have significant differences.

C. Fuzzy Controller

The fuzzy controller is used to perform decision-making via a fuzzy inference process. The inference process involves the operation of fuzzy logic, fuzzy rules and fuzzy membership functions. The inputs and outputs of the fuzzy controller are descripted in linguistic terms with fuzzy membership functions. The fuzzy controller can perform decision-making based on human knowledge by using fuzzy logic and fuzzy rules.

To implement the fuzzy inference system, there are two major types of implementation: Mamdani method and the Sugeno method. In the proposed intelligent optimiser, the Mamdani method is adopted to perform fuzzy inference for determining the WM-DE parameters. For the Mamdani method, the inputs and outputs are described by fuzzy sets. To realise making decision, the linguistic control rules in fuzzy controller are designed by experienced human experts. At the final stage of decision-making in the fuzzy controller, a process called defuzzification is used to generate the crisp output values from the corresponding fuzzy sets and corresponding membership functions. In the proposed intelligent optimiser, the defuzzification is realised by the method of centre of gravity (COG).

For the population-based searching algorithm like WM-DE, because of the randomness of the searching process, different experiments of a given problem may have different movement of the population. With different population distribution information in different stages of the searching process, we could have different system parameter values for the WM-DE engines. The way of choosing those system parameter values could be described as some heuristic rules. As a result, a fuzzy controller can be employed to determine the F and C_r values. The t-value of the current generation and the standard deviations of the population fitness from the two WM-

DE engines can act as the input for the fuzzy controller to determine the F and C_r values for the next generation. In the proposed intelligent optimiser, the fuzzy controller act as a closed-loop control system to vary the parameter values of the two WM-DE engines at different searching states.

A fuzzy controller processes linguistic variables, and each variable normally consists of a set of linguistic terms. In the proposed intelligent optimiser, the linguistic terms are modelled by triangular-shaped membership functions, which can be written as:

$$f_m\left(x_m(t), a_m, b_m, c_m\right) = \begin{cases} 0, & x_m(t) \le a_m \\ \dfrac{x_m(t) - a_m}{b_m - a_m} & a_m \le x_m(t) \le b_m \\ \dfrac{c_m - x_m(t)}{c_m - b_m} & b_m \le x_m(t) \le c_m \\ 0, & c_m \le x_m(t) \end{cases} \tag{4.5}$$

where $x_m(t)$ is the input of the fuzzy membership function, m=1, 2,…,m_f; m_f denotes the number of membership functions. Fig. 4.3 represents three triangular-shaped membership functions (of 3 fuzzy terms M1, M2 and M3) for an input variable. The input range is between zero and one. Every input and output of the fuzzy controller in the proposed intelligent optimiser has three linguistic terms that cover different values of input and output. The reason for choosing the triangular-shaped membership function is that it is less computational demanding than other types of membership functions. The triangular-shaped membership function requires simple arithmetic operations only.

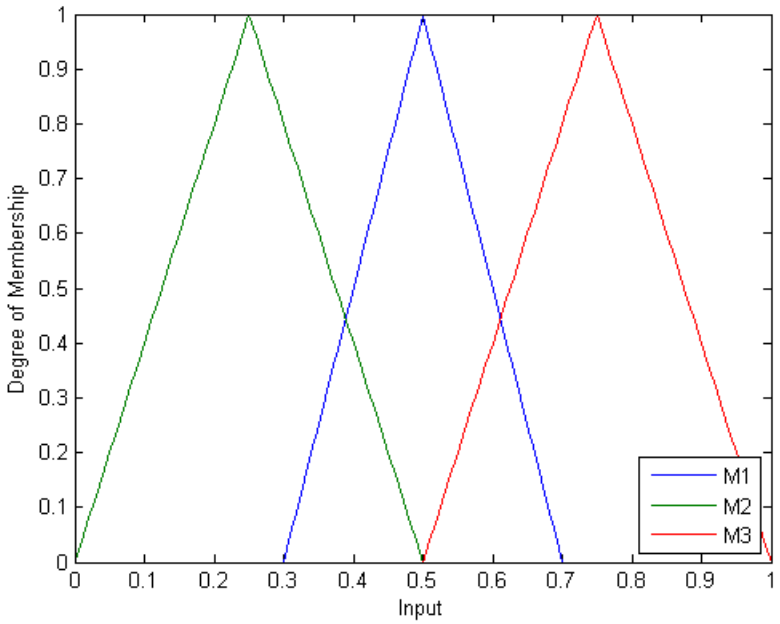


Fig. 4.3. Triangular-shaped membership functions.

In the proposed system, the membership functions for the three inputs are $f_{tg,m}(t_g, a_{tg,m}, b_{tg,m}, c_{tg,m})$, $f_{\sigma g1,m}(\sigma_{g1}, a_{\sigma g1,m}, b_{\sigma g1,m}, c_{\sigma g1,m})$ and $f_{\sigma g2,m}(\sigma_{g2}, a_{\sigma g2,m}, b_{\sigma g2,m}, c_{\sigma g2,m})$. And the four outputs are $f_{Fg1,m}(F_{g1}, a_{Fg1,m}, b_{Fg1,m}, c_{Fg1,m})$, $f_{Fg2,m}(F_{g2}, a_{Fg2,m}, b_{Fg2,m}, c_{Fg2,m})$, $f_{Crg1,m}(C_{rg1}, a_{Crg1,m}, b_{Crg1,m}, c_{Crg1,m})$ and $f_{Crg2,m}(C_{rg2}, a_{Crg2,m}, b_{Crg2,m}, c_{Crg2,m})$. In the inference process, the fuzzy controller generates the fuzzified output by processing the fuzzified inputs with a set of fuzzy if-then rules. There are four formats of rules in the fuzzy rule base, one for each output:

Rule τ:	**IF**	t_g	is	$f_{tg,m}(t_g, a_{tg,m}, b_{tg,m}, c_{tg,m})$
	AND	σ_{g1}	is	$f_{\sigma g1,m}(\sigma_{g1}, a_{\sigma g1,m}, b_{\sigma g1,m}, c_{\sigma g1,m})$
	THEN	F_{g1}	is	$f_{Fg1,m}(F_{g1}, a_{Fg1,m}, b_{Fg1,m}, c_{Fg1,m})$
Rule τ:	**IF**	t_g	is	$f_{tg,m}(t_g, a_{tg,m}, b_{tg,m}, c_{tg,m})$
	AND	σ_{g2}	is	$f_{\sigma g2,m}(\sigma_{g2}, a_{\sigma g2,m}, b_{\sigma g2,m}, c_{\sigma g2,m})$
	THEN	F_{g2}	is	$f_{Fg2,m}(F_{g2}, a_{Fg2,m}, b_{Fg2,m}, c_{Fg2,m})$
Rule τ:	**IF**	t_g	is	$f_{tg,m}(t_g, a_{tg,m}, b_{tg,m}, c_{tg,m})$
	AND	σ_{g1}	is	$f_{\sigma g1,m}(\sigma_{g1}, a_{\sigma g1,m}, b_{\sigma g1,m}, c_{\sigma g1,m})$
	THEN	C_{rg1}	is	$f_{Crg1,m}(C_{rg1}, a_{Crg1,m}, b_{Crg1,m}, c_{Crg1,m})$
Rule τ:	**IF**	t_g	is	$f_{tg,m}(t_g, a_{tg,m}, b_{tg,m}, c_{tg,m})$
	AND	σ_{g2}	is	$f_{\sigma g2,m}(\sigma_{g2}, a_{\sigma g2,m}, b_{\sigma g2,m}, c_{\sigma g2,m})$
	THEN	C_{rg2}	is	$f_{Crg2,m}(C_{rg2}, a_{Crg2,m}, b_{Crg2,m}, c_{Crg2,m})$

The rule number is denoted by τ, where $\tau = 1,2, \ldots, n_r$; n_r is the total number of rules in the fuzzy rule base.

After the fuzzification of the inputs, aggregation of each rule output is performed to generate an output as a single fuzzy set. The output of aggregation for the corresponding fuzzy rule is defined as follows:

$$o_\tau = \min(IN_{1,\tau}, IN_{2,\tau}) \tag{4.6}$$

where $IN_{1,\tau}$ and $IN_{2,\tau}$ are the two input membership function values of the fuzzy rule τ. The fuzzy output of each rule is realised as follows.

$$OUT_\tau = o_\tau \cdot OUTPUT_\tau \tag{4.7}$$

where OUT_τ is the output membership function of the corresponding rule and $OUTPUT_\tau$ is the defined output membership function.

During the inference process, we map all the output membership functions from all rules to form the aggregated result for performing defuzzification. The process of defuzzification is

used to determinate the numerical output of the fuzzy system. In the intelligent optimiser, the defuzzification is implemented by using the Centroid method, which uses the centroid point of the aggregated result as the numerical result. The Centroid defuzzification is given by,

$$y(t) = \frac{\int OUT_{all} \cdot ydy}{\int OUT_{all} dy} \tag{4.8}$$

where OUT_{all} is the aggregated result of all the corresponding rules, and $y(t)$ is the corresponding numerical output of the fuzzy system.

In the proposed intelligent optimiser, the membership functions for the inputs and outputs of the internal fuzzy controller are shown in Fig. 4.4 and Fig. 4.5. The design of the membership functions is based on many experiments with the knowledge of the characteristics of DE. For the input of normalised standard deviation, the range is between 0 and 0.2. We consider the value smaller than 0.05 as LOW and the value larger than 0.15 as HIGH. For the input of *t*-value, the range is between 0 and 10. For, $\xi = 50$, if the *t*-value is equal or higher than 1.645, we could have the confidence that those two engines' populations have significant difference between each other. As a result, we consider the *t*-value smaller than 1.645 as LOW.

For the outputs of F and C_r, they are ranged between 0 and 0.9. In most of the application, DE works well with the values of F and C_r between 0.3 and 0.5. We consider the value of F and C_r as LOW if they are smaller than 0.1, and we consider the value of F and C_r as HIGH if they are larger than 0.7.

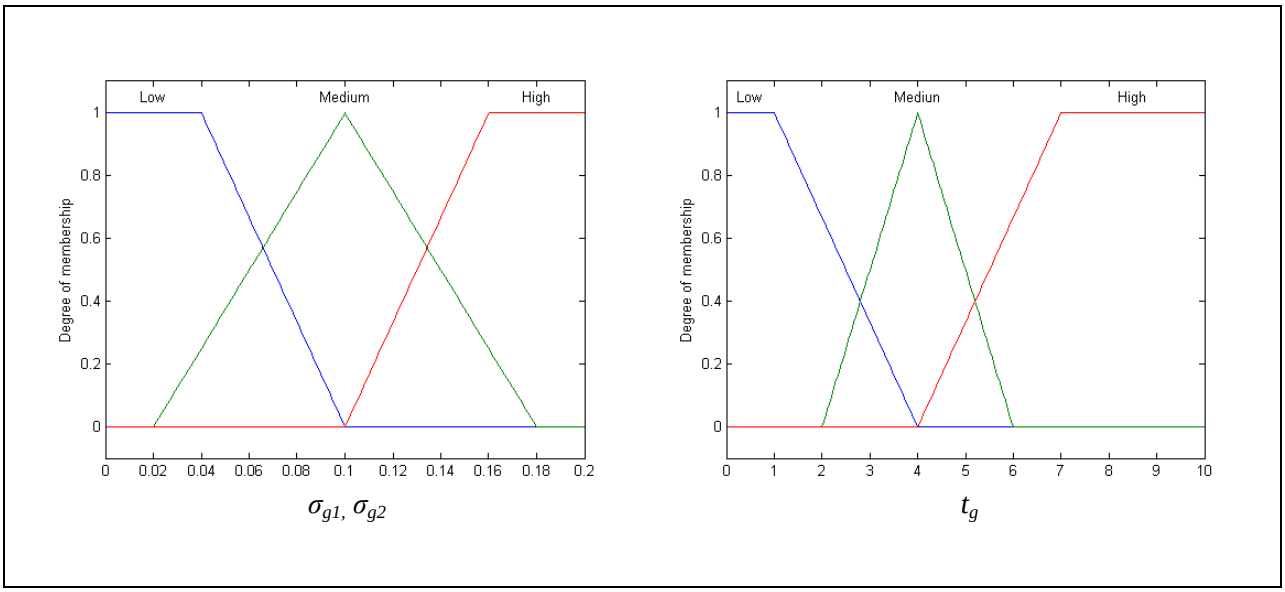


Fig. 4.4. Input membership functions.

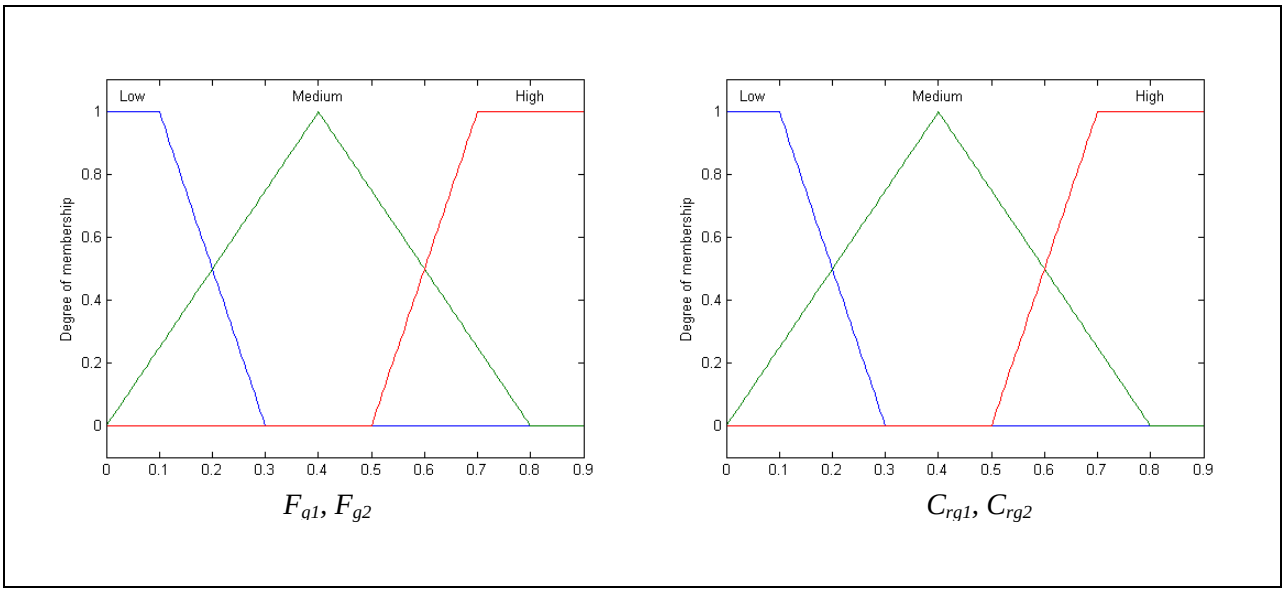


Fig. 4.5. Output membership functions.

Totally 36 fuzzy rules can be generated. They are:

For the F parameter:

1. IF (σ_{g1} is low)	AND (t_g is low)	THEN (F_{g1} is low)
2. IF ($\sigma_{g1}$is medium)	AND (t_g is low)	THEN (F_{g1} is low)
3. IF (σ_{g1} is high)	AND (t_g is low)	THEN (F_{g1} is medium)
4. IF (σ_{g1} is low)	AND (t_g is medium)	THEN (F_{g1} is medium)
5. IF (σ_{g1} is medium)	AND (t_g is medium)	THEN (F_{g1} is medium)
6. IF (σ_{g1} is high)	AND (t_g is medium)	THEN (F_{g1} is high)
7. IF (σ_{g1} is low)	AND (t_g is high)	THEN (F_{g1} is medium)
8. IF (σ_{g1} is medium)	AND (t_g is high)	THEN (F_{g1} is medium)
9. IF (σ_{g1} is high)	AND (t_g is high)	THEN (F_{g1} is high)
10. IF (σ_{g2} is low)	AND (t_g is low)	THEN (F_{g2} is low)
11. IF (σ_{g2} is medium)	AND (t_g is low)	THEN (F_{g2} is low)
12. IF (σ_{g2} is high)	AND (t_g is low)	THEN (F_{g2} is medium)
13. IF (σ_{g2} is low)	AND (t_g is medium)	THEN (F_{g2} is medium)
14. IF (σ_{g2} is medium)	AND (t_g is medium)	THEN (F_{g2} is medium)
15. IF (σ_{g2} is high)	AND (t_g is medium)	THEN (F_{g2} is high)
16. IF (σ_{g2} is low)	AND (t_g is high)	THEN (F_{g2} is medium)
17. IF (σ_{g2} is medium)	AND (t_g is high)	THEN (F_{g2} is medium)
18. IF (σ_{g2} is high)	AND (t_g is high)	THEN (F_{g2} is high)

For the C_r parameter:

1. IF (σ_{g1} is low)	AND (t_g is low)	THEN (C_{rg1} is low)
2. IF (σ_{g1} is medium)	AND (t_g is low)	THEN (C_{rg1} is low)
3. IF (σ_{g1} is high)	AND (t_g is low)	THEN (C_{rg1} is medium)
4. IF (σ_{g1} is low)	AND (t_g is medium)	THEN (C_{rg1} is medium)
5. IF (σ_{g1} is medium)	AND (t_g is medium)	THEN (C_{rg1} is medium)
6. IF (σ_{g1} is high)	AND (t_g is medium)	THEN (C_{rg1} is high)
7. IF (σ_{g1} is low)	AND (t_g is high)	THEN (C_{rg1} is medium)
8. IF (σ_{g1} is medium)	AND (t_g is high)	THEN (C_{rg1} is medium)
9. IF (σ_{g1} is high)	AND (t_g is high)	THEN (C_{rg1} is high)
10. IF (σ_{g2} is low)	AND (t_g is low)	THEN (C_{rg2} is low)
11. IF (σ_{g2} is medium)	AND (t_g is low)	THEN (C_{rg2} is low)
12. IF (σ_{g2} is high)	AND (t_g is low)	THEN (C_{rg2} is medium)
13. IF (σ_{g2} is low)	AND (t_g is medium)	THEN (C_{rg2} is medium)
14. IF (σ_{g2} is medium)	AND (t_g is medium)	THEN (C_{rg2} is medium)
15. IF (σ_{g2} is high)	AND (t_g is medium)	THEN (C_{rg2} is high)
16. IF (σ_{g2} is low)	AND (t_g is high)	THEN (C_{rg2} is medium)
17. IF (σ_{g2} is medium)	AND (t_g is high)	THEN (C_{rg2} is medium)
18. IF (σ_{g2} is high)	AND (t_g is high)	THEN (C_{rg2} is high)

The first of 18 rules are used for the F parameter and the second 18 rules are used for the C_r parameter. In SDE, the F parameter is very important for the mutation operation that produces a new vector for the next generation. On doing the DE mutation operation, the difference of two vectors is calculated and then multiplied by F. Then, it is added to one of the vectors in the population to obtain a new vector for the next generation. The F parameter controls the displacement from the old vector to the new vector. A large value of F means the new vector will move far away from the original position. The basic design principle of the fuzzy rules is that we want the searching to have a balance between the exploration and exploitation of the searching space. In the early stage of searching, we want more exploration while more exploitation is desired at the later stage of searching. By examining the population standard deviation of the fitness values, we can estimate the progress of searching of the WM-DE engine. In practice, the population is randomly distributed in the solution space in the early stage of the searching process, making the standard deviation to be large and the value of F to be high. It moves towards the global optimal point along the searching at the later stage, making the standard deviation to be small and the value of F to be low.

With the null hypothesis that there should not have any significant difference between the two WM-DE engines as they are solving the same objective function, the t-value should tend to be small along the searching. The rules for the F parameter are governed by two inputs: the population standard deviation of the fitness values and the t-value. The two inputs are encoded with three linguistic terms. As a result, nine rules can be formulated. Since there are two WM-DE engines in the intelligent optimiser and controlled by the fuzzy controller, and the control operation are based on the corresponding population standard deviation of fitness values of one engine, totally 18 rules for the F parameter can be formulated.

Similarly, another 18 rules can be formulated for the C_r parameter. The C_r parameter affects DE on doing the crossover operation, which modifies the vector elements after the DE mutation operation. It can be considered as introducing some random disturbances to the DE operation. Similar to the F parameter setting, we want the searching to have a balance between the exploration and exploitation of the searching space. In the early stage of searching, we want more exploration while more exploitation is desired at the later stage of searching. If the value of C_r is large, it means that there is a bigger difference between the resulting vector and the original vector. To have more exploration in the early stage and more exploitation in the later stage, we want the value of C_r to be large in the early stage of the searching process. As a result, the population can have more chance to move away from some local optima. In the later stage of searching, we want the value of C_r to be small to do a fine-tuning for the final solution. With the null hypothesis that there should not have any significant difference between the two WM-DE engines as they are using the same objective function, the t-value should tend to be small along the searching.

The block diagram of the internal fuzzy controller is shown in Fig. 4.6. This fuzzy controller has 3 inputs and 4 outputs, and generates outputs for every iteration step.

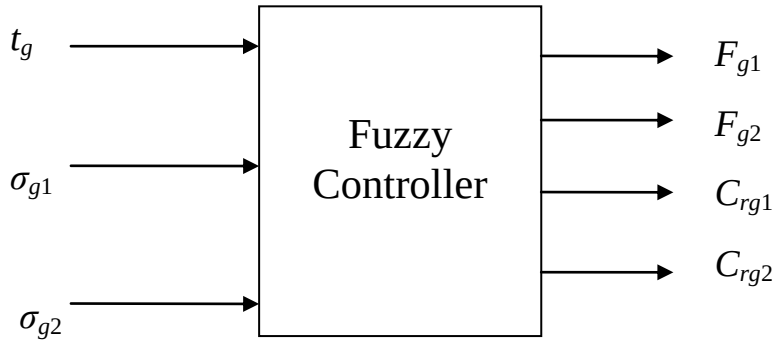


Fig. 4.6. Inputs and outputs of the fuzzy controller.

D. Wavelet-mutated Differential Evolution (WM-DE)

The Wavelet-mutated Differential Evolution (WM-DE) algorithm is a simplified version of Differential Evolution with Double Wavelet Mutations (DWM-DE) algorithm. In WM-DE, only one wavelet mutation is taken after the DE crossover operation. Fig. 4.7 shows the pseudo code for the WM-DE. In the DWM-DE, the weighting F is generated by a wavelet function. In the intelligent optimiser, the weighting F is generated by the fuzzy controller.

```
begin
Initialise the population
    while (not termination condition) do
     begin
       Mutation operation by equation (3.2)
       Crossover operation by equation (3.3)
       Modifying the trail population vectors based on equation (3.15)
       Evaluation of the fitness functions
     end
end
```

Fig. 4.7. Pseudo code of WM-DE.

III BENCHMARK TEST FUNCTIONS AND RESULTS

A. Benchmark Test Functions

In Chapter 3, a suite of 29 benchmark test functions is used to test the performance of the DWM-DE. In this Chapter, the same suite of benchmark test functions is also used to evaluate the performance of the intelligent optimiser. As discussed in Chapter 3, these 29 benchmark test functions covers many different kinds of optimisation problems and can be separated into four main categories. The functions f_1 - f_8 (first category) are unimodal functions that involve a symmetric solution space and contain a single optimum point only. The functions f_9 - f_{16} are multimodal functions with a few local minima; they are put to the second category. The third category covers the multimodal functions with many local minima; functions f_{17} - f_{24} belong to this category. The last category contains functions with shift and rotate; functions f_{25} - f_{29} belong to this category, which are the shifted and rotated versions of some functions of the pervious categories. The list of the benchmark test functions are shows in Table 3.1 and the definitions of these functions are listed in Table 3.2.

B. Experimental Setup

We evaluate the performance of SDE, DWM-DE (discussed in Chapter 3), DE/local-to-best/1, DE/rand/1 with per-vector-dither and the proposed intelligent optimiser. These five optimisation methods are employed to find the minimum values of the benchmark test functions. The list below shows the simulation conditions of the experiments:

- Shape parameter of the wavelet mutation (ζ_{wm}) (for DWM-DE and the proposed intelligent optimiser): 1
- Parameter λ for the monotonic increasing function (for DWM-DE and the proposed intelligent optimiser): 10000.
- Initial population: It is generated uniformly at random.
- Mutation scaling factor (for SDE, DE/local-to-best/1 and DE/rand/1): $F = 0.5$
- Crossover rate factor (for DWM-DE, SDE, DE/local-to-best/1 and DE/rand/1): $C_r = 0.5$
- Population size: 50
- Numbers of iteration for all algorithms: As listed in Table 4.1

Table 4.1. Maximum Number of Iteration.

Test Function	No. of Iteration
f_1. Sphere model	500
f_2. Generalised Rosenbrock's function	5000
f_3. Step function	100
f_4. Quartic function	200
f_5. Schwefel's problem 2.21	500
f_6. Schwefel's problem 2.22	500
f_7. Easom's function	500
f_8. McCormick function	50
f_9. Shekel's foxholes function	50
F_{10}. Kowalik's function	100
f_{11}. Maxican hat function	50
f_{12}. Six-hump camel back function	50
f_{13}. Hartman's family 1	50
f_{14}. Hartman's family 2	100
f_{15}. Egg Holder function	1000
f_{16}. Styblinski-Tang function	50
f_{17}. Generalised penalised function	200
f_{18}. Generalised Rastrigin's function	1000
f_{19}. Generalised Griewank's function	500
F_{20}. Ackley's function	500
F_{21}. Schwefel's function	500
f_{22}. Schaffer function	500
f_{23}. Chichinadze function	100
f_{24}. Sine Envelope Sine Wave function	10000
f_{25}. Shifted Sphere model	500
f_{26}. Shifted Schwefel's problem 1.2	500
f_{27}. Shifted rotated high conditioned elliptic function	200
f_{28}. Shifted Rosenbrock's function	200
f_{29}. Shifted Rastrigin's function	1000

C. Results and Discussion

In this experiment, all results reported for the benchmark test functions are averaged ones out of 50 trials.

1. Unimodal functions

Function f_1 has a smooth and symmetric surface around the solution. It is a model of sphere. Due to its smooth surface, most of the methods can converge to the global minimum but at different rates. Fig. 4.8(a) shows the convergence rates. It shows that the proposed intelligent optimiser could provide the best performance in terms of convergence rate. In terms of mean cost value and best cost value, the proposed intelligent optimiser offers better result than the other methods as shown in Table 4.2. In addition, the standard deviation is small, which means that the proposed intelligent optimiser offer a reliable searching mechanism. By introducing the fuzzy control and the T-Test measurement in the proposed intelligent optimiser, the population distribution information can be studied by the optimiser to control the internal parameters of F and C_r during the searching. Thanks to the T-Test measurement, we could help a stable progress of evolution. This leads to better solution reliability reached by the proposed intelligent optimiser.

Function f_2 is the Generalised Rosenbrock's function. It has a smooth and symmetric surface around the solution. The function shape looks like a "Banana". The solution space of this function contains a flat gorge. Similar to f_1, the main purpose of testing is to measure the convergence rate of the searching methods. The result is shown in Fig 4.8(b). The proposed intelligent optimiser offers the highest convergence rate. When using the proposed intelligent optimiser, the solution quality is increased. In terms of mean value and standard deviation as

shown in Table 4.2, the intelligent optimiser and DWM-DE performs better than the other methods.

Function f_3 is a Step function. Most of the optimisation algorithms failed to handle the functions with flat surface because the optimisation algorithms could not obtains any searching direction from the flat surface. Function f_3 is one of the functions with many flat surfaces. All the methods involve in this study could handle this function well as shown in Fig 4.8(c) and Table 4.2. Thanks for the adaptive control inside the proposed intelligent optimiser, better solution quality, solution convergence rate and solution reliability can be obtained.

Function f_4 is the Quartic function. It contains a global minimum at the centre of the solution space. The experiment results are shown in Fig 4.8(d) and Table 4.2. The convergence rate offered by the proposed intelligent optimiser is much higher than that of the other DE methods. After around 10 times of iteration, the proposed intelligent optimiser is able to reach the minimum.

Functions f_5 is the Schwefel's problem 2.21. The 4.8(e) shows the experiment results. Functions f_6 is the Schwefel's problem 2.22. The experiment results are shown in Fig. 4.8(f). The convergence rate for functions f_5 and f_6 of the proposed intelligent optimiser is the highest. The best solution, mean and standard derivation provided by the intelligent optimiser are the best as shown in Table 4.2.

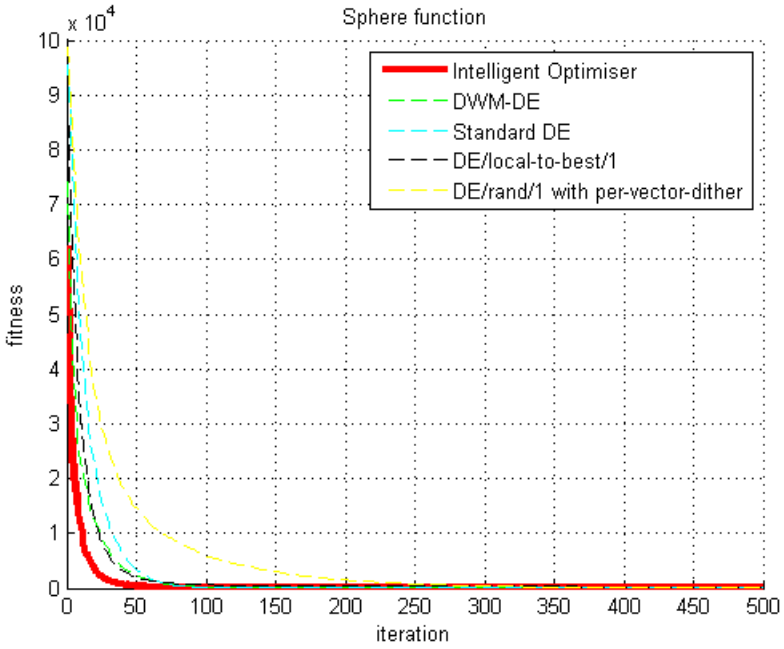
Function f_7 is the Easom function. The function was inverted for minimisation. A very large solution space for this function is designed to test the performance of the proposed intelligent optimiser. The result is shown in Fig. 4.8(g). The convergence rate for functions f_7 of

the proposed intelligent optimiser is the highest. It gives better performance in terms of solution quality and reliability as shown in Table 4.2.

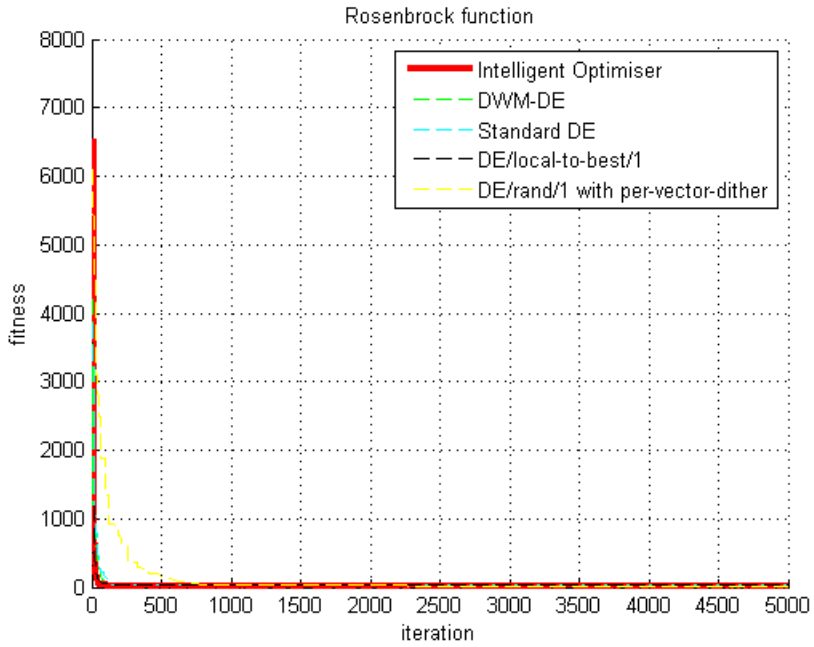
Function f_8 is the McCormick function, which is a two-dimension benchmark function with a global minimum at $f(-0.54719, -1.54719) = -1.9133$. The McCormick function contains a flat and smooth search surface. The experimental results show that the intelligent optimiser and DWM-DE can provide the best solution quality and the best solution reliability as shown in Table 4.2. Moreover, Fig. 4.8(h) shows that the proposed intelligent optimiser can provide the highest rate of convergence.

Table 4.2. Comparison between Different DE Methods for Benchmark Test Functions (Category 1).

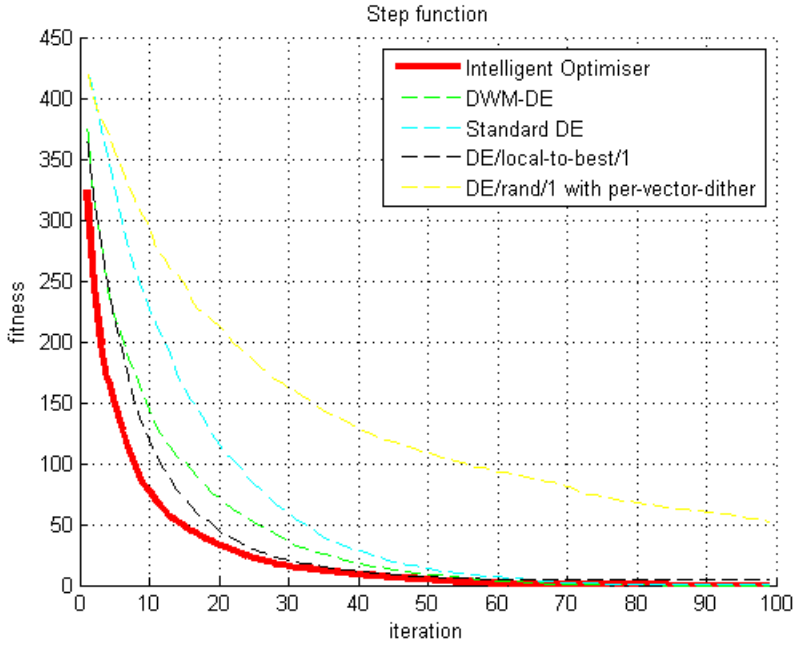
		intelligent optimiser	DWM-DE	SDE	DE/ local-to-best/ 1	DE/rand/ 1 with per-vector -dither
f_1	Mean	0.035028	0.5902	0.9937	228.8271	411.8185
	Best	0.000377	0.0605	0.4317	17.3954	206.3942
	Std Dev	0.031695	0.5712	0.308	213.2461	99.2895
f_2	Mean	0.0002	0.0961	25.3632	40.3851	30.2437
	Best	0.0002	0.0068	23.7534	26.9933	27.3151
	Std Dev	0	0.0867	0.6442	15.6161	3.9049
f_3	Mean	0	0	11.24	4.7	51.54
	Best	0	0	7	1	35
	Std Dev	0	0	1.9119	3.4241	8.1346
f_4	Mean	0.0294	0.0385	0.2307	0.5176	4.2939
	Best	0.0144	0.0172	0.1033	0.0798	1.7894
	Std Dev	0.0091	0.0111	0.0684	0.3172	1.3556
f_5	Mean	1.2127	1.4127	5.5862	44.3662	22.7523
	Best	0.6033	0.7089	3.7069	21.1518	19.1513
	Std Dev	0.3432	0.3512	2.9702	9.0817	2.05
f_6	Mean	0.0285	0.388	3.3391	4.3577	27.5979
	Best	0.0031	0.1151	2.3578	0.1878	20.7384
	Std Dev	0.0250	0.187	0.5352	3.6601	3.2958
f_7	Mean	-1	-1	-0.9284	-0.66	-0.4641
	Best	-1	-1	-1	-1	-1
	Std Dev	0	0	0.2488	0.4785	0.4454
f_8	Mean	-1.913223	-1.913223	-1.913199	-1.913223	-1.913152
	Best	-1.913223	-1.913223	-1.91322	-1.913223	-1.913219
	Std Dev	0	0	0.000023	0	0.00007


Sphere function
x 10^4
fitness
iteration
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

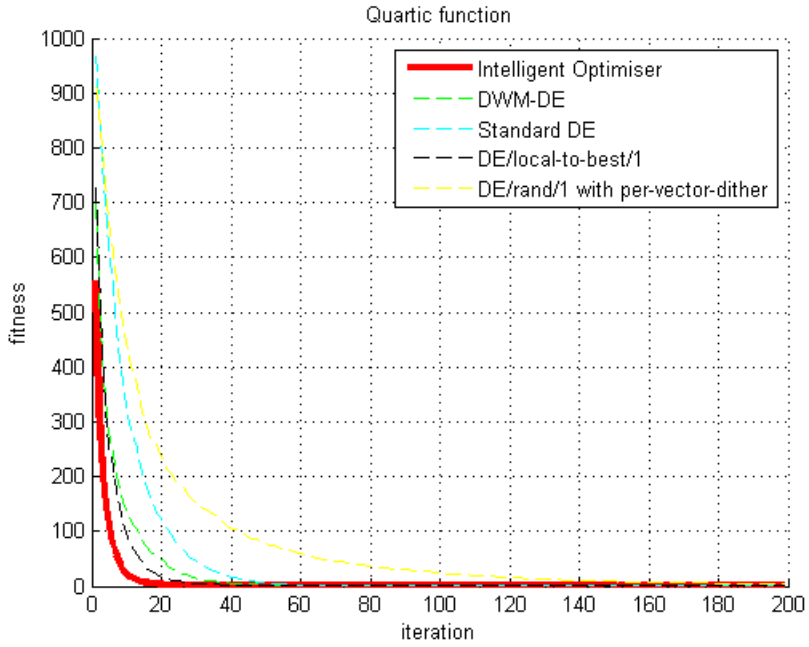
(a)


Rosenbrock function
fitness
iteration
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

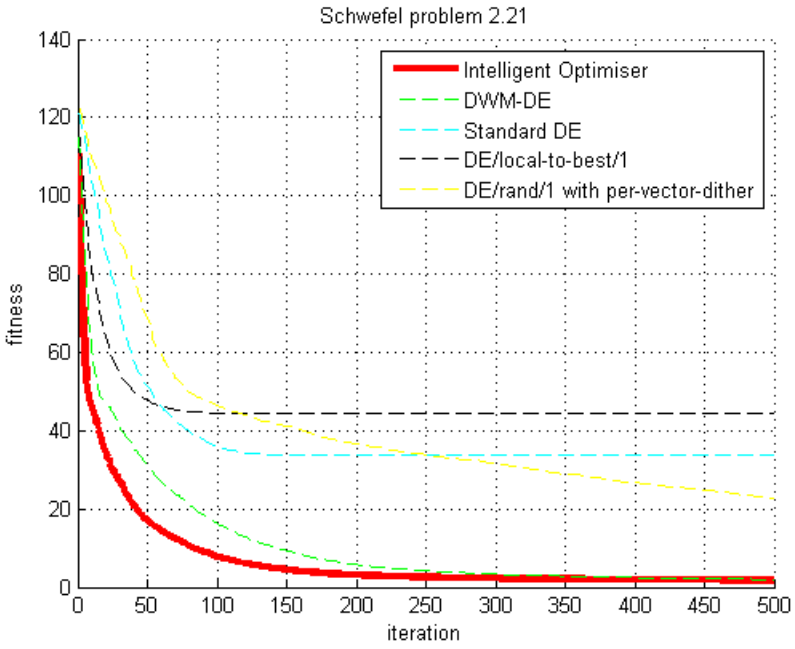
(b)


Step function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
50
100
150
200
250
300
350
400
450
0
10
20
30
40
50
60
70
80
90
100

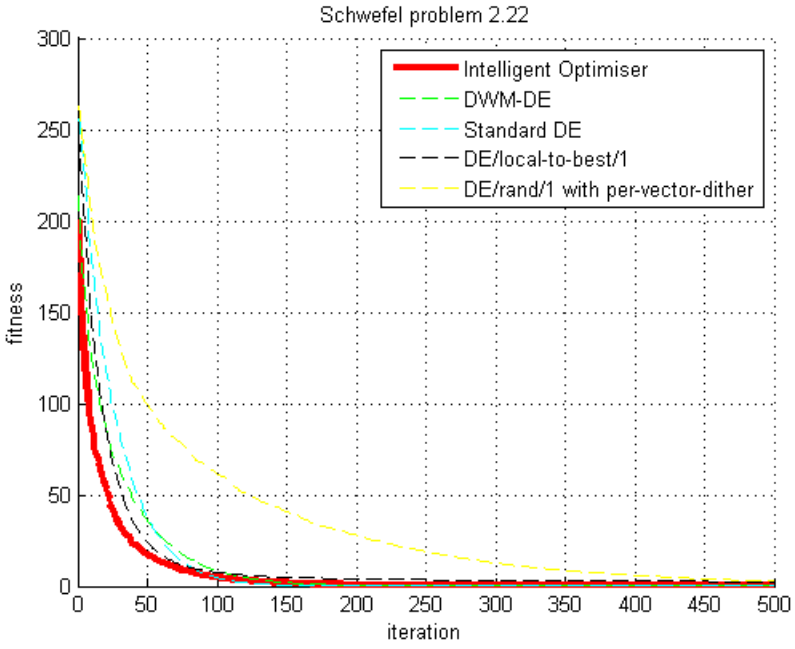
(c)


Quartic function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
100
200
300
400
500
600
700
800
900
1000
0
20
40
60
80
100
120
140
160
180
200

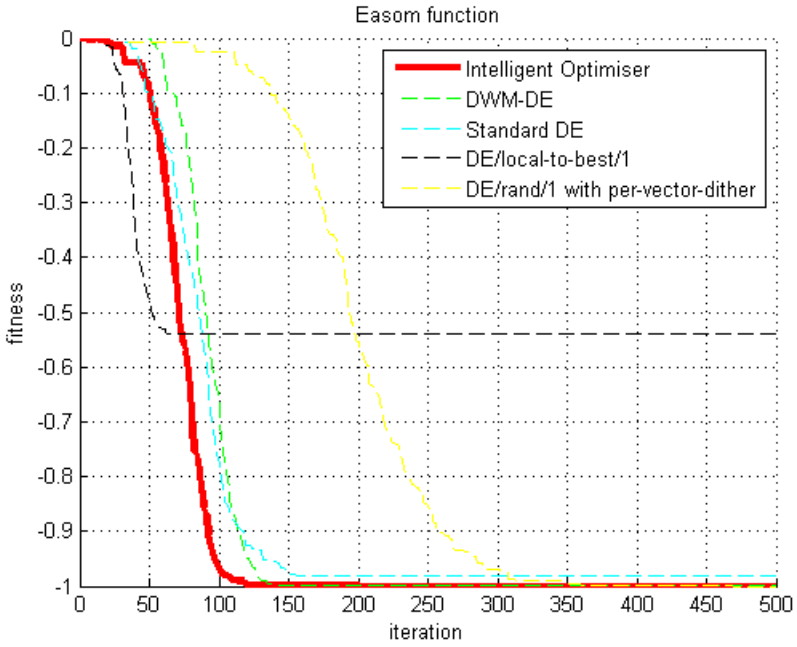
(d)


Schwefel problem 2.21
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
20
40
60
80
100
120
140
0
50
100
150
200
250
300
350
400
450
500

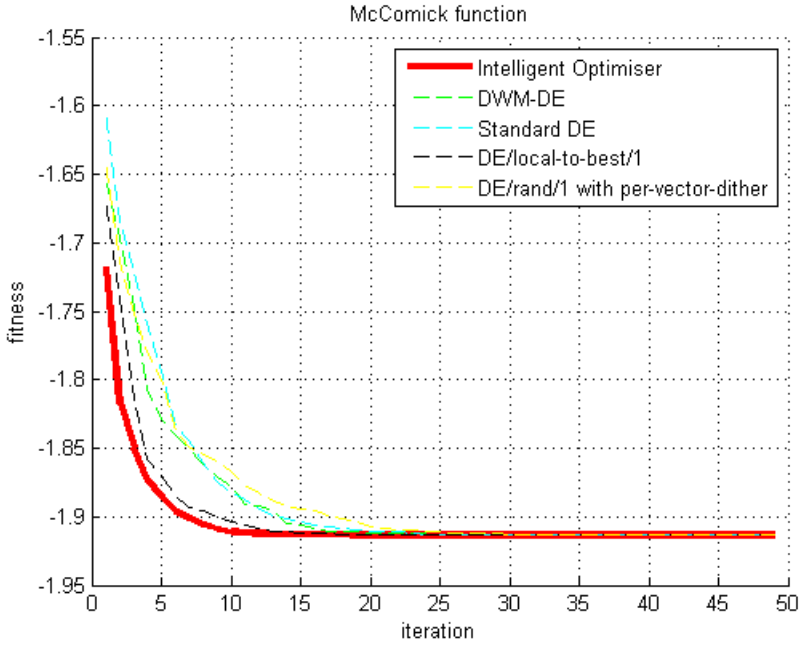
(e)


Schwefel problem 2.22
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
50
100
150
200
250
300
0
50
100
150
200
250
300
350
400
450
500

(f)



(g)



(h)

Fig. 4.8. Results for unimodal functions.

For unimodal functions, the proposed intelligent optimiser and DWM-DE can offer a higher rate of convergence. More exploration would be done by the mutation operations during the early stage of evolution. In the later stage of evolution, the fine-tuning ability of the wavelet operations leads to more exploitation of the small region around the global minimum. Besides, by introducing the fuzzy control and the T-Test measurement in the proposed intelligent optimiser, the population distribution information can be studied by the optimiser to control the values of the internal parameters F and C_r during the searching. As a result, better solution quality and solution convergence rate can be obtained. Moreover, the small standard deviation values of the solutions show that the T-Test measurement could help a stable progress of searching within the two populations. This leads to better solution reliability of the proposed intelligent optimiser. In short, the proposed intelligent optimiser is the best to tackle unimodal functions among the DE methods covered in Table 4.2.

2. Multimodal functions with a few local minima

Eight multimodal functions with a few local minima are used to evaluate the five DE methods. Those functions are the Shekel's foxholes function, Kowalik's function, Maxican hat function, Six-hump camel back function, Hartman's family 1 function, Hartman's family 2 function, Egg holder function and Styblinski-Tang function. The experimental results for these functions are listed in Table 4.3 and shown in Fig. 4.9.

For functions f_9 to f_{14}, as discussed in Chapter 3, the experimental results show that all the optimisation methods involved in the experiment offer similar results on searching the optimal point. Although the functions contain a few local minima, no trapping in the local minima is

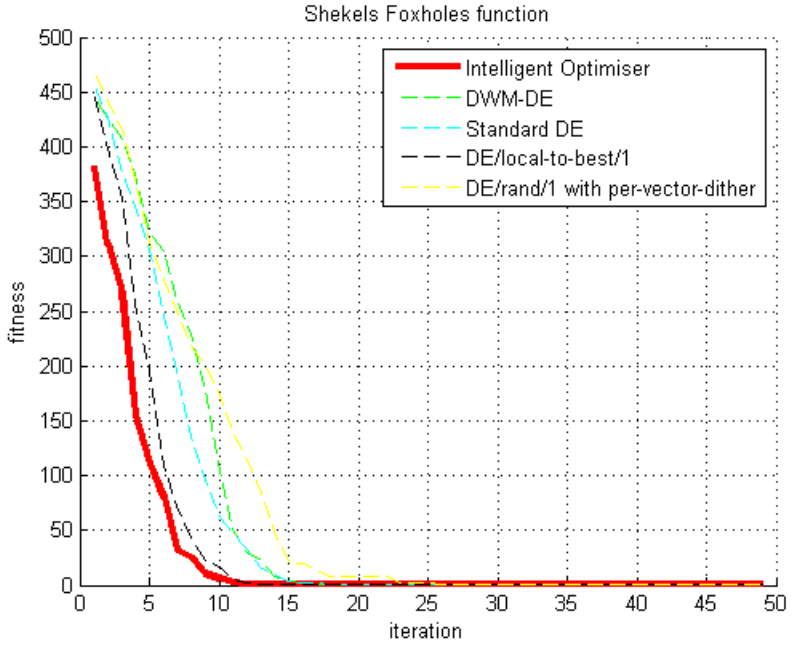
found in all the methods. Yet, the proposed intelligent optimiser performs better than the other methods in term of solution quality, convergence rate, and solution reliability.

Function f_{15} is the Egg-holder function, which is a well-established benchmark test function for evaluating the performance of global optimisation algorithms. It is a difficult test function to be optimised, especially when the dimension is high (>20). The optimiser could be trapped in the local optima easily. In the experiment, we set the Egg-holder function's dimension to 20. In terms of convergence speed, solution quality and solution reliability, the experimental results show that the intelligent optimiser performs better than the other methods. Meanwhile, when the dimension of the Egg-holder function is two, the intelligent optimiser just performs nearly the same as the conventional methods. This test function demonstrates that when the dimension of the problem is high, the effect of the wavelet mutation becomes more significant.

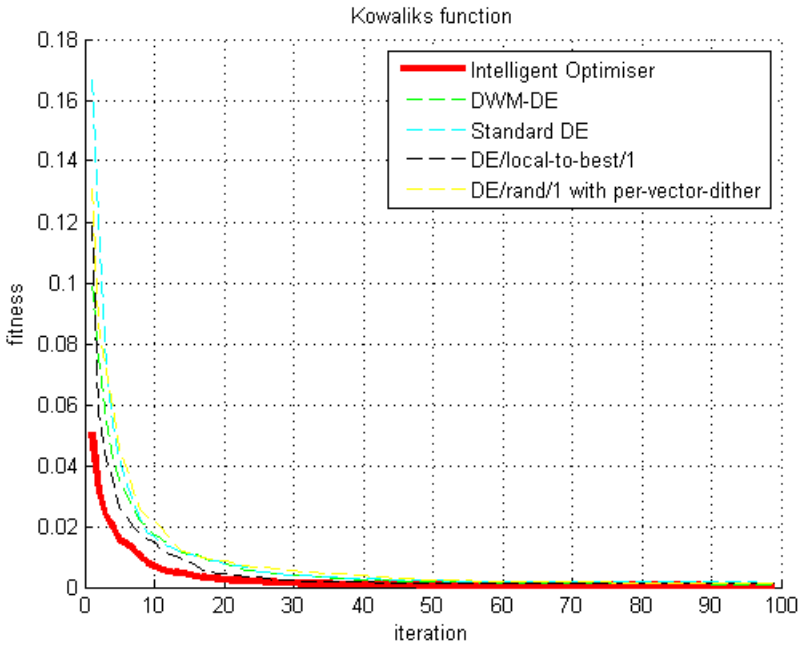
Function f_{16} is the Styblinski-Tang function. It is a 2-dimension benchmark function with the global minimum at f (–0.903534, –0.903534) = –78.332. This function has a bowl shape surface with four separated regions in which each region contains one local minimum. Moreover, one of the four separated regions contains the global minimum. Owing to the nature of four separated regions, an optimiser could not recover easily if it is trapped in one of the regions in the searching domain. The experimental results are shown in Fig. 4.9(h). The convergence rate of the proposed intelligent optimiser is the highest. Both the proposed intelligent optimiser and DWM-DE give better performance in terms of solution quality and reliability as shown in Table 4.3.

Table 4.3. Comparison between Different DE Methods for Benchmark Test Functions (Category 2).

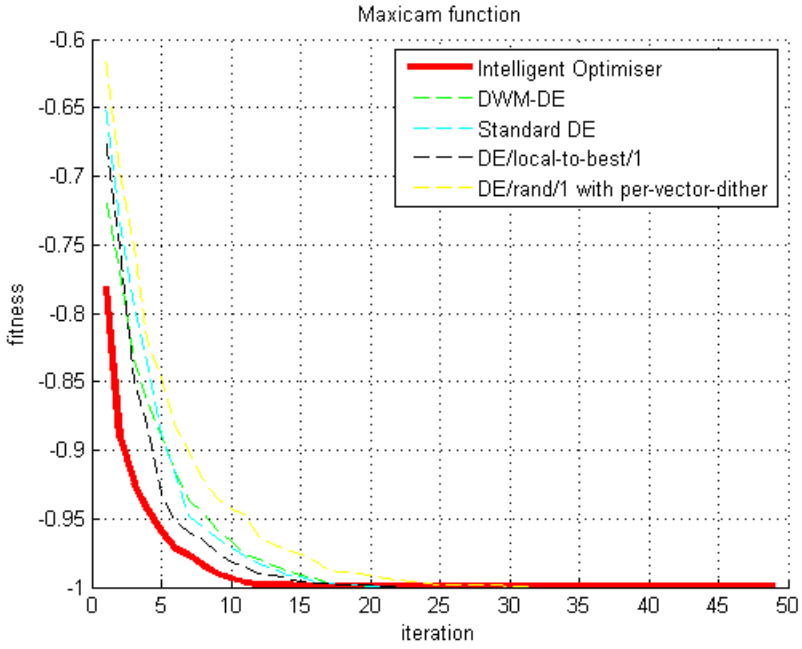
		Intelligent Optimiser	DWM-DE	SDE	DE/ local-to-best/ 1	DE/rand/1 with per-vector -dither
f_9	Mean	0.998	0.998	0.998	0.998	0.998
	Best	0.998	0.998	0.998	0.998	0.998
	Std Dev	0	0	0	0	0
f_{10}	Mean	0.0005	0.0009	0.0011	0.0015	0.0016
	Best	0.0003	0.0005	0.0007	0.0003	0.0007
	Std Dev	0.0001	0.0003	0.0008	0.0039	0.0018
f_{11}	Mean	-1	-1	-1	-1	-1
	Best	-1	-1	-1	-1	-1
	Std Dev	0	0	0	0	0
f_{12}	Mean	-1.0316	-1.0316	-1.0316	-1.0316	-1.0315
	Best	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	Std Dev	0	0	0.0001	0	0.0002
f_{13}	Mean	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628
	Best	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628
	Std Dev	0	0	0	0	0
f_{14}	Mean	-3.3199	-3.3124	-3.2909	-3.2911	-3.2777
	Best	-3.3229	-3.322	-3.322	-3.322	-3.322
	Std Dev	0.0300	0.0303	0.0475	0.0527	0.0448
f_{15}	Mean	-26008.6	-25462.7	-21094.9	-11467	-10975.7
	Best	-26333.5	-26333.5	-23402	-17668.3	-13185.6
	Std Dev	213.2864	546.7434	1093.533	1949.098	763.1555
f_{16}	Mean	-78.332331	-78.332331	-78.33233	-78.33233	-78.332318
	Best	-78.332331	-78.332331	-78.332331	-78.332331	-78.332331
	Std Dev	0.000001	0.000001	0.000003	0.000003	0.000018


Shekels Foxholes function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
50
100
150
200
250
300
350
400
450
500
0
5
10
15
20
25
30
35
40
45
50

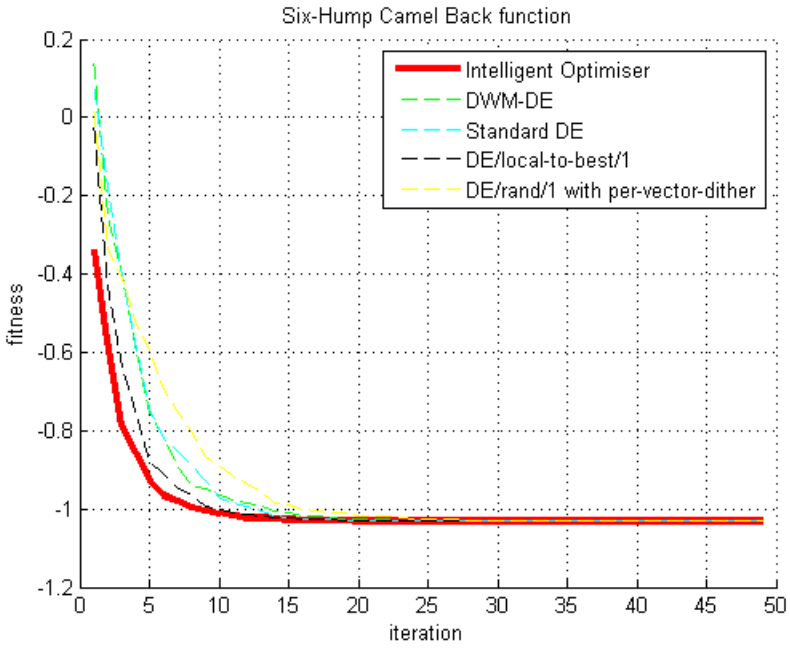
(a)


Kowaliks function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0
10
20
30
40
50
60
70
80
90
100

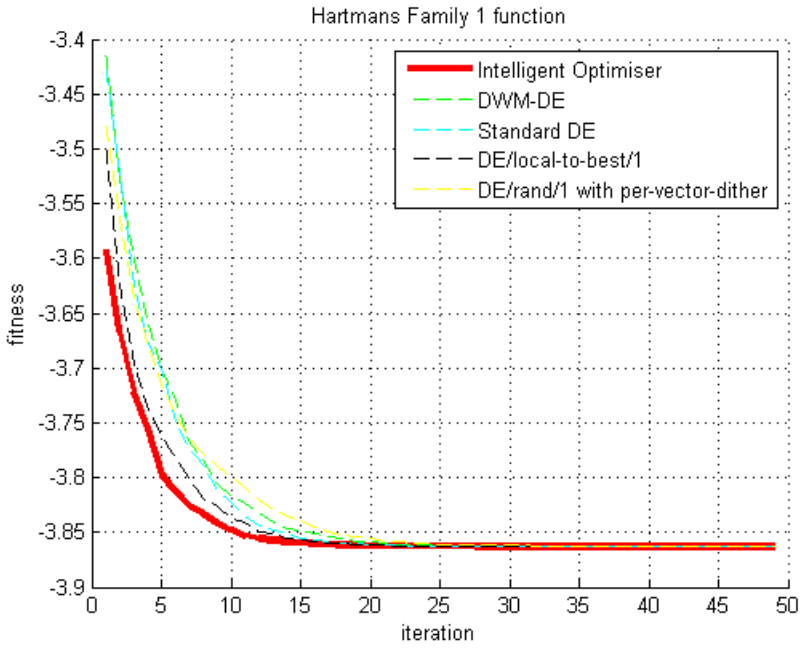
(b)


Maxicam function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
-0.6
-0.65
-0.7
-0.75
-0.8
-0.85
-0.9
-0.95
-1
0
5
10
15
20
25
30
35
40
45
50

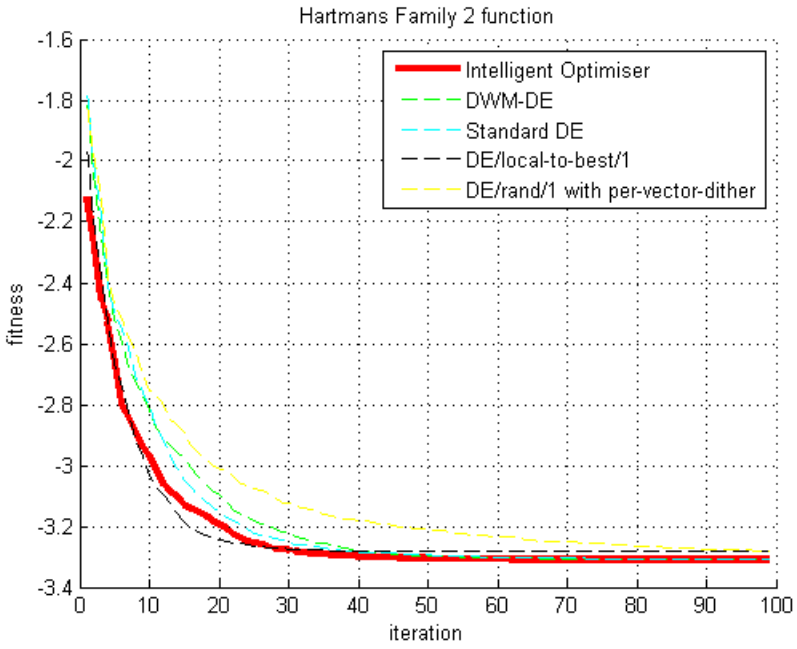
(c)


Six-Hump Camel Back function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
5
10
15
20
25
30
35
40
45
50

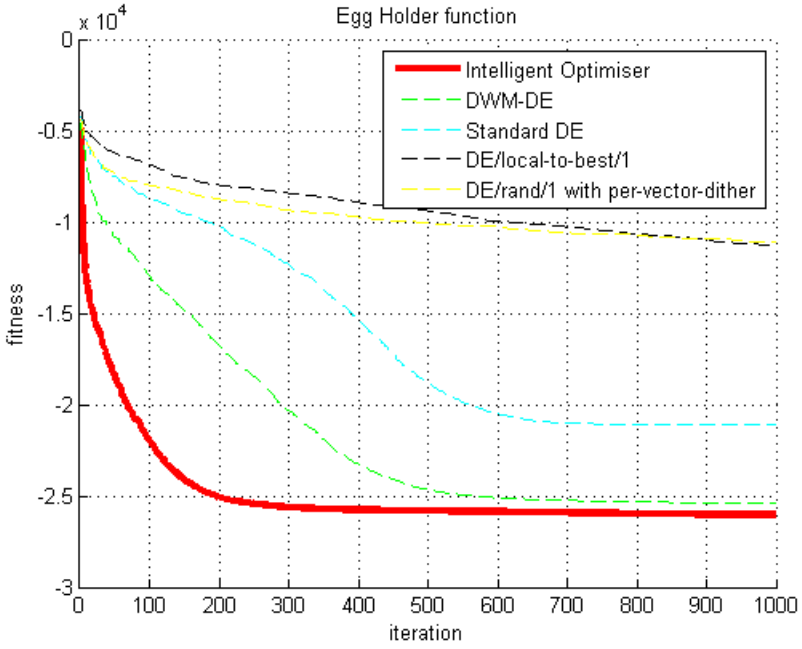
(d)


Hartmans Family 1 function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
-3.4
-3.45
-3.5
-3.55
-3.6
-3.65
-3.7
-3.75
-3.8
-3.85
-3.9
0
5
10
15
20
25
30
35
40
45
50

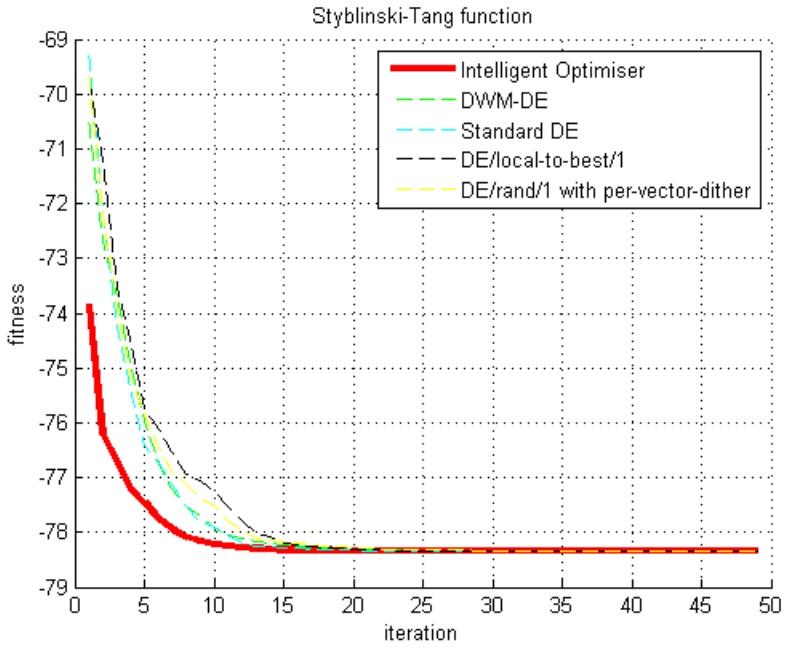
(e)


Hartmans Family 2 function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
-1.6
-1.8
-2
-2.2
-2.4
-2.6
-2.8
-3
-3.2
-3.4
0
10
20
30
40
50
60
70
80
90
100

(f)



(g)



(h)

Fig. 4.9. Results for multimodal functions with a few local minima.

For multimodal functions with a few local minima, the proposed intelligent optimiser and DWM-DE can perform better on providing a higher rate of convergence. More exploration could be done by taking advantage of the mutation operation. During the later stage of evolution, the population can exploit the small region around the global minimum. Besides, by introducing the fuzzy control and the T-Test measurement in the proposed intelligent optimiser, the population distribution information can be used by the optimiser to adjust the values of the internal parameters F and C_r adaptively during the searching. As a result, better solution quality and convergence rate can be obtained. Moreover, thanks to the T-Test measurement, we could realise a stable progress of searching within the two populations. This leads to better solution reliability offered by the proposed intelligent optimiser. It is reflected by the small standard deviation values of the solutions for the test functions. In short, the proposed intelligent optimiser is the best to tackle multimodal functions with a few local minima when compared with the other methods.

3. Multimodal functions with many local minima

Functions f_{17} - f_{24} are multimodal functions with many local minima. The experimental results for these functions are listed in Table 4.4 and shown in Fig. 4.10. Function f_{17}, f_{18}, f_{19} are the Generalised penalised function, Generalised Rastrigin's function and Generalised Griewank's function respectively. They are widely used as test functions for global optimisation algorithms. Those functions have a lot of local minima distributed regularly. When the function dimension increased, the number of local minima increases exponentially. In the experiments, the function contains plenty of local minima as the dimension is 30. From Fig. 4.10, we can see that if the

proposed intelligent optimiser is used, the rate of convergence is improved. Moreover, by introducing the fuzzy control and the T-Test measurement in the proposed intelligent optimiser, the population distribution information can be used by the optimiser to control the values of the internal parameters F and C_r adaptively during the searching. As a result, better solution quality and convergence rate can be obtained. With the support of the T-Test measurement, we could realise a more stable progress of searching within the two populations. This leads to better solution reliability offered by the proposed intelligent optimiser.

Function f_{20} is the Generalised Ackley's function. It modulates a cosine wave into an exponential function. It is a continuous multimodal function with a flatland and a central minimum. The result is shown in Fig. 4.10(d). It shows that the proposed intelligent optimiser and the DWM-DE converge rapidly. After 250 times of iteration, the population has nearly reached the global minimum. It shows the advantage of the wavelet mutation operation on reducing the number of steps for moving and the evaluation to those regions containing local minima but far away from the global minimum.

Function f_{21} is the Schwefel's function. Most of the optimisation algorithms fail to handle this function because the global minimum is isolated from the similar best local minimum geometrically. The population is potentially prone to converge to the local minimum. The result is shown in Fig. 4.10(e). Similar to functions f_{19} and f_{20}, if the proposed intelligent optimiser is used, the convergence rate is much improved. It can nearly reach the global minimum at round 100 times of iteration.

Function f_{22} is the Schaffer function. It is a 2-dimension benchmark function that has a single global minimum at the point $f_{22}(0,0) = 0$. It contains a large number of local minima.

Searching for the global minimum is a difficult task because the value at the best local minimum and the global minimum differs by about 0.0001 only. From the experiments, although the convergence rate is not significantly improved, the intelligent optimiser can offer better solution reliability and better solution quality as shown in Table 4.4.

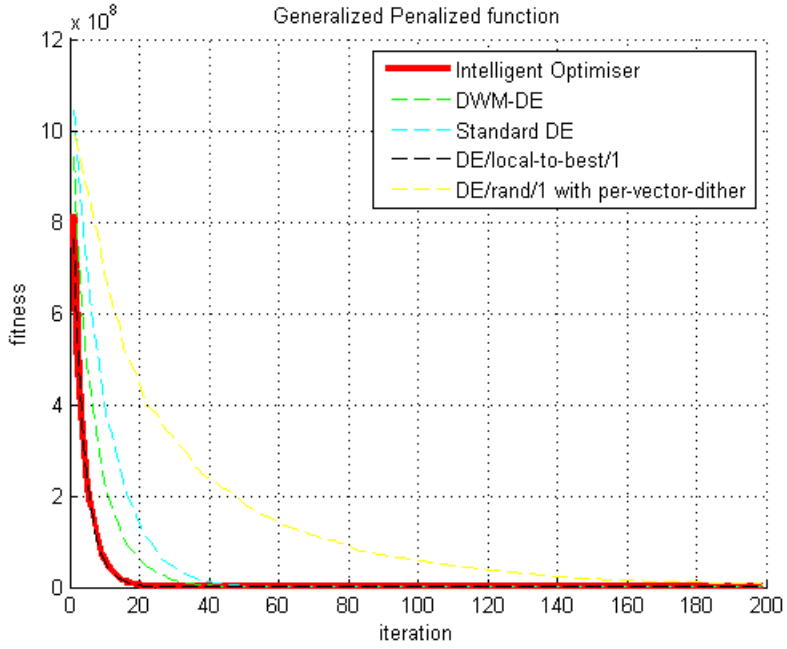
Function f_{23} is the Chichinadze function, which is a 2-dimension benchmark function with a global minimum at $f(5.90133, 0.5) = -43.3259$. As the nature of this function is not complex, most of the methods offer the same result. However, the proposed intelligent optimiser shows the highest convergence rate.

Function f_{24} is the Sine envelope sine wave function. It is also called the Schaffer function. The Sine envelope sine wave function is a multi-dimensional version of the Schaffer function. It contains a large number of local minima. Searching for the global minimum is a difficult task because the difference between the value at the best local minimum and the global minimum is about 0.001 only. The number of local optima is not well defined, and they are continuously spread around the global optimum. Theoretically, there are infinite local minima that form a number of grooves around the global minimum. In the experiment, we set the Sine envelope sine wave function with a dimension of 20. In terms of solution quality and solution reliability, the experimental results show that the intelligent optimiser performs better than the other methods, although the intelligent optimiser could not offer the highest convergence rate.

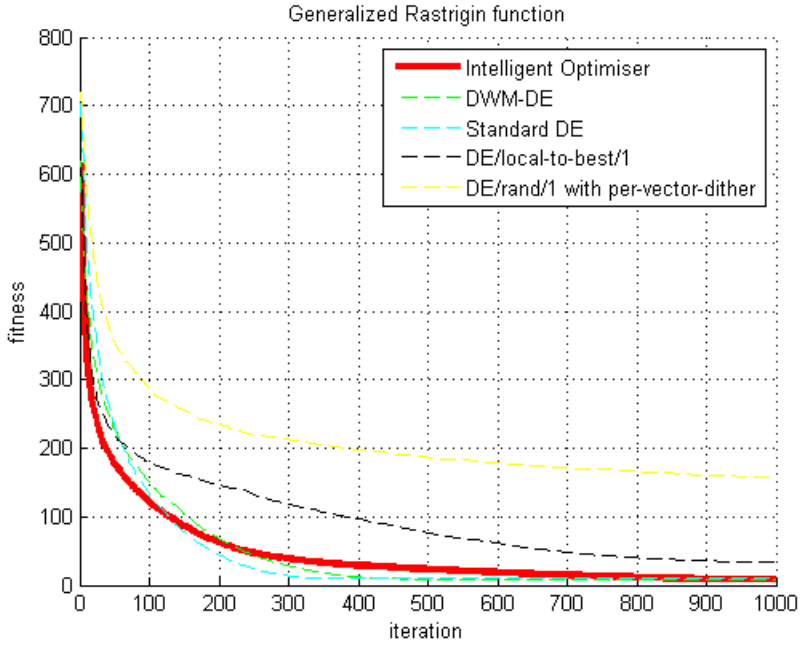
Table 4.4. Comparison between Different DE Methods for Benchmark Test Functions (Category 3).

		intelligent optimiser	DWM-DE	SDE	DE/ local-to-best/ 1	DE/rand/1 with per-vector -dither
	Mean	0.4346	0.4883	207.3875	147.6601	6835946

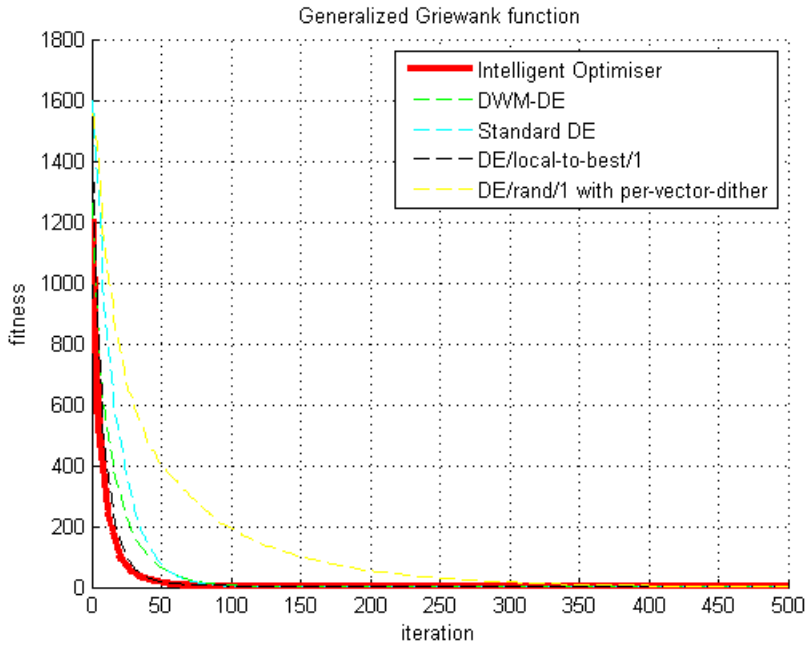
f_{17}	Best	0.1308	0.1582	22.4585	4.7027	149.9788
	Std Dev	0.1999	0.2989	291.9504	391.5459	450.1213
f_{18}	Mean	8.1902	8.3213	124.7772	34.9675	160.9108
	Best	0.000067	0.0063	87.5478	16.9349	136.2236
	Std Dev	3.2217	4.4707	10.5849	11.9889	10.8145
f_{19}	Mean	0.181280	1.0243	2.2483	3.0293	54.1213
	Best	0.016792	0.8751	1.5661	1.0394	33.3442
	Std Dev	0.099791	0.0576	0.4224	1.9004	10.991
f_{20}	Mean	0.029157	0.3199	0.7723	4.6762	17.728
	Best	0.014373	0.0271	0.0261	2.2245	7.1453
	Std Dev	0.012089	0.3194	2.9323	1.5461	3.687
f_{21}	Mean	-12569.468	-12568.8	-12475.8	-10328.1	-10128.1
	Best	-12569.481	-12569.5	-12569	-11004.5	-11272.4
	Std Dev	0.010201	0.6883	145.1123	391.6365	560.2403
f_{22}	Mean	0.003886	0.005249	0.006801	0.000759	0.002625
	Best	0	0	0	0.000001	0
	Std Dev	0.004808	0.004889	0.004498	0.001983	0.003478
f_{23}	Mean	-43.315862	-43.315756	-43.315719	-43.315739	-43.310539
	Best	-43.315862	-43.315859	-43.315866	-43.315859	-43.315487
	Std Dev	0	0.000219	0.000187	0.000221	0.005238
f_{24}	Mean	0.7858	0.9907	1.0441	4.0761	7.7249
	Best	0.4602	0.6446	0.6390	3.5609	6.8711
	Std Dev	0.2617	0.2868	0.4637	0.5406	0.5424


Generalized Penalized function
x 10^8
fitness
iteration
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

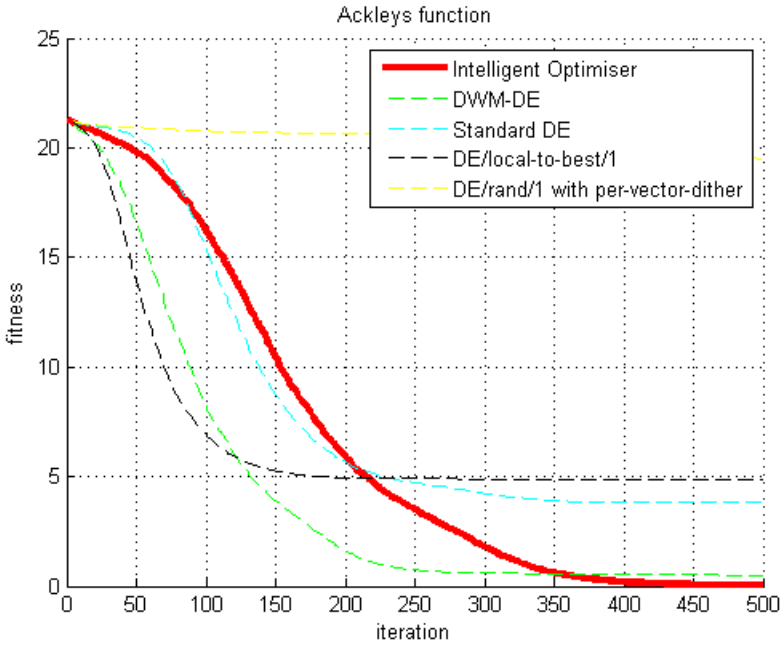
(a)


Generalized Rastrigin function
fitness
iteration
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

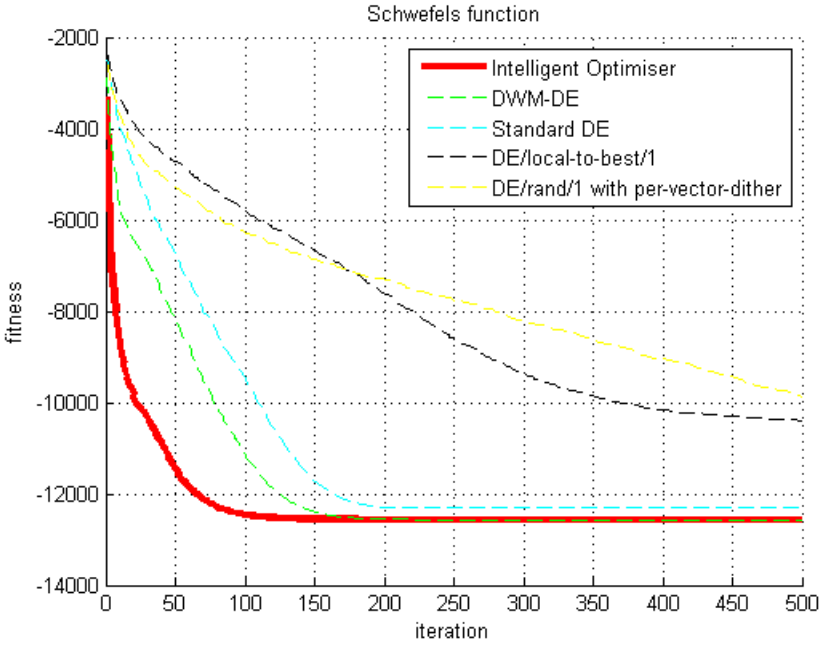
(b)


Generalized Griewank function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
200
400
600
800
1000
1200
1400
1600
1800
0
50
100
150
200
250
300
350
400
450
500

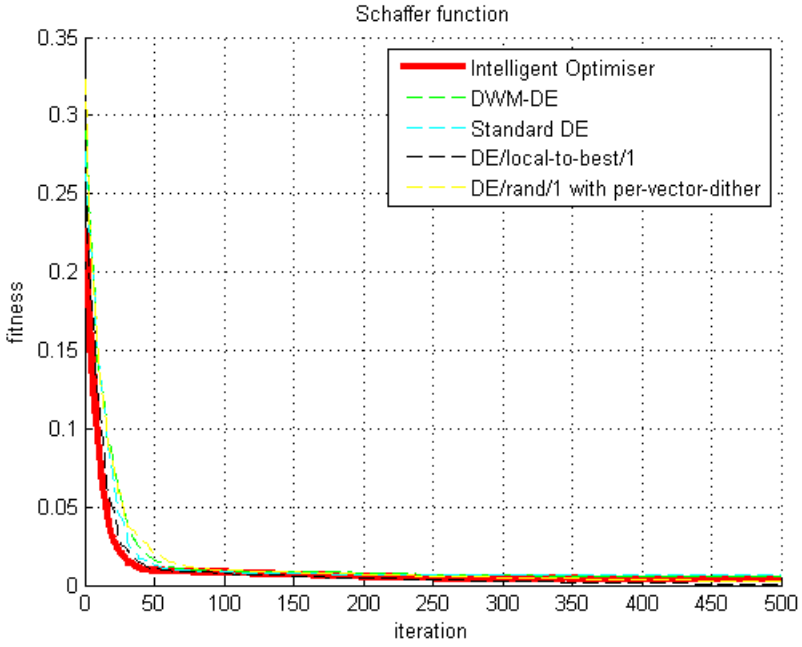
(c)


Ackleys function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0
5
10
15
20
25
0
50
100
150
200
250
300
350
400
450
500

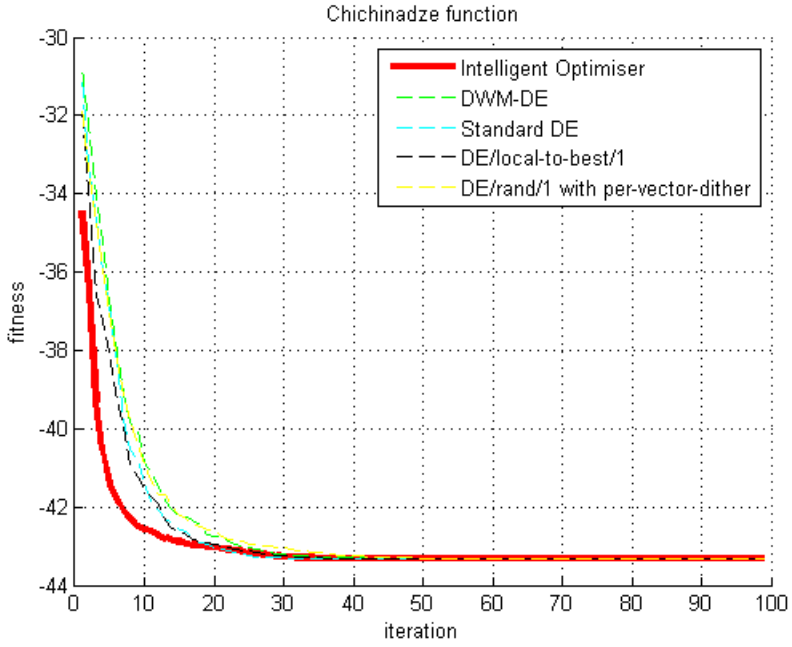
(d)


Schwefels function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
-2000
-4000
-6000
-8000
-10000
-12000
-14000
0
50
100
150
200
250
300
350
400
450
500

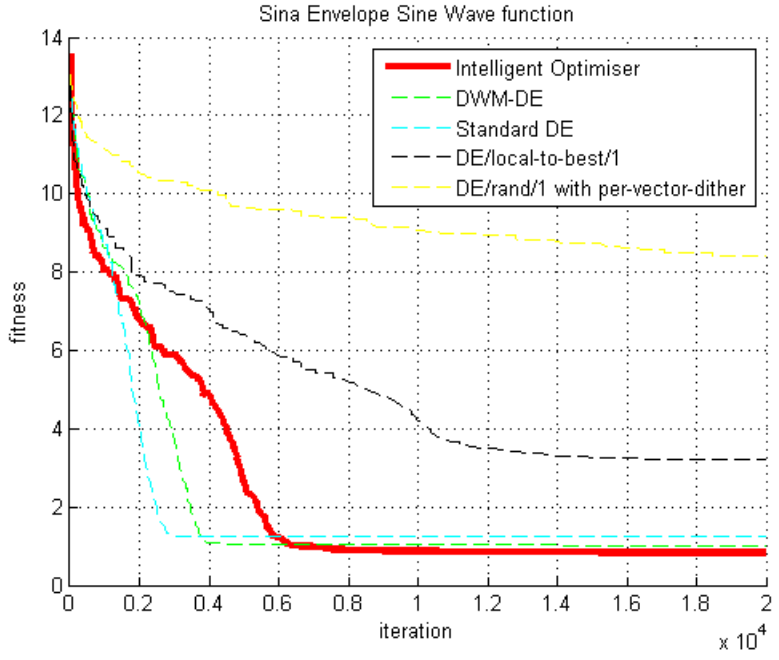
(e)


Schaffer function
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
fitness
iteration
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
50
100
150
200
250
300
350
400
450
500

(f)



(g)



(h)

Fig. 4.10. Results for multimodal functions with many local minima.

In general, compared to the other methods, the proposed intelligent optimiser can significantly improve the chance of reaching the global optimum and the rate of convergence for multimodal functions with many local minima.

4. Functions with shift and rotate

Five functions with shift and rotate are employed for testing the performance of the intelligent optimiser. Three functions are unimodal functions. They are the Shifted sphere model, Shifted Schwefel's problem 1.2, and Shifted rotated high conditioned elliptic function. Two functions are multimodal functions. They are the Shifted Rosenbrock's function, and Shifted Rastrigin's function.

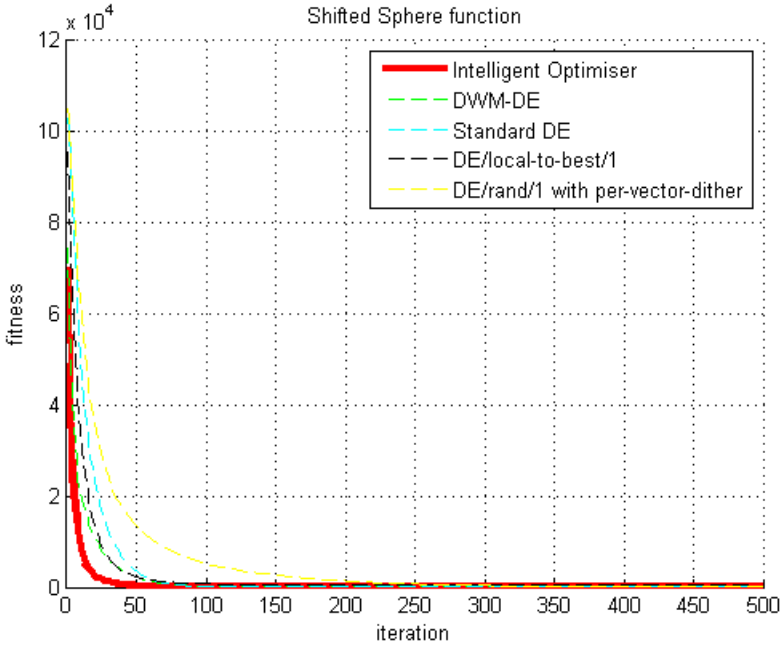
Some of the benchmark functions in the previous three categories have drawbacks as the elements in the optimum vector might have the same value for different dimensions owing to the symmetry nature. The global optimum is normally located at the centre of the searching domain. Some optimisers are designed to converge to the centre of the searching domain even no searching direction is provided. Hence, this kind of benchmark functions might not be good to evaluate the real performance of optimisers. To overcome these drawbacks, shift and rotate are introduced to the benchmark test functions to generate optimum points with different numerical values. As a result, the optimum point is not lying at the centre of the searching domain.

The experimental results for functions with shift and rotate are listed in Table 4.5 and shown in Fig. 4.11. The results show that the proposed intelligent optimiser still can offer the best performance. The advantage brought by the sharing of population information of the two embedded WM-DE engines and the adaptive control of the internal parameter values of the

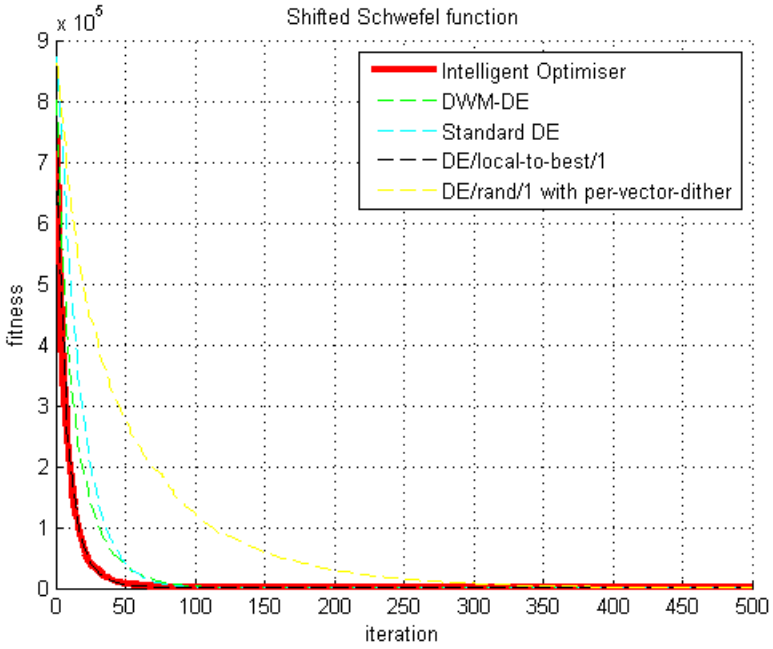
optimiser could enhance the searching performance in all functions of this category. The solution quality and reliability offered by intelligent optimiser are good.

Table 4.5. Comparison between Different DE Methods for Benchmark Test Functions (Category 4).

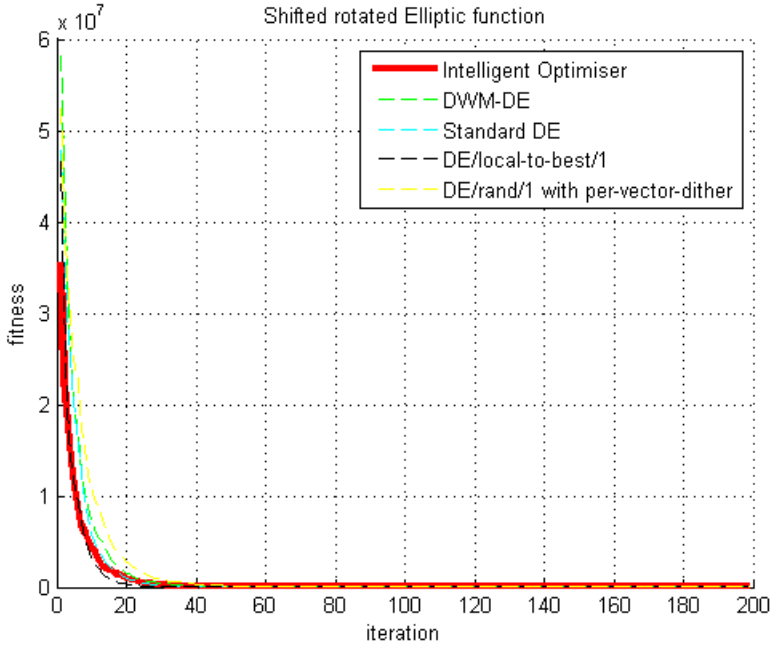
		intelligent optimiser	DWM-DE	SDE	DE/ local-to-best/ 1	DE/rand/1 with per-vector -dither
f_{25}	Mean	0.00036	0.000742	41.30417	349.7417	26.59024
	Best	0.000119	0.000244	0.019934	7.378115	13.61648
	Std Dev	0.000294	0.000302	135.404	389.9865	10.21125
f_{26}	Mean	0.005599	0.011645	199.2697	374.9674	575.92
	Best	0.007681	0.002418	0.123243	3.066111	259.5999
	Std Dev	0.002605	0.004996	466.9468	429.4013	163.9056
f_{27}	Mean	0.00112	0.00149	194.3431	0.004001	2.516334
	Best	0.000046	0.000056	0.000341	0.000096	0.396908
	Std Dev	0.001003	0.001327	815.1713	0.028289	1.447321
f_{28}	Mean	1.113245	4.760516	5.435293	4.099365	11.69429
	Best	0.000085	0.001795	0.009373	0.00047	0.008133
	Std Dev	1.809608	8.102966	8.352652	6.416724	15.96312
f_{29}	Mean	7.114673	8.267052	10.32804	35.18819	159.3997
	Best	1.22332	2.004883	5.175221	12.95799	127.034
	Std Dev	1.75345	2.98555	3.015084	10.97512	10.40491


Shifted Sphere function
x 10^4
fitness
iteration
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

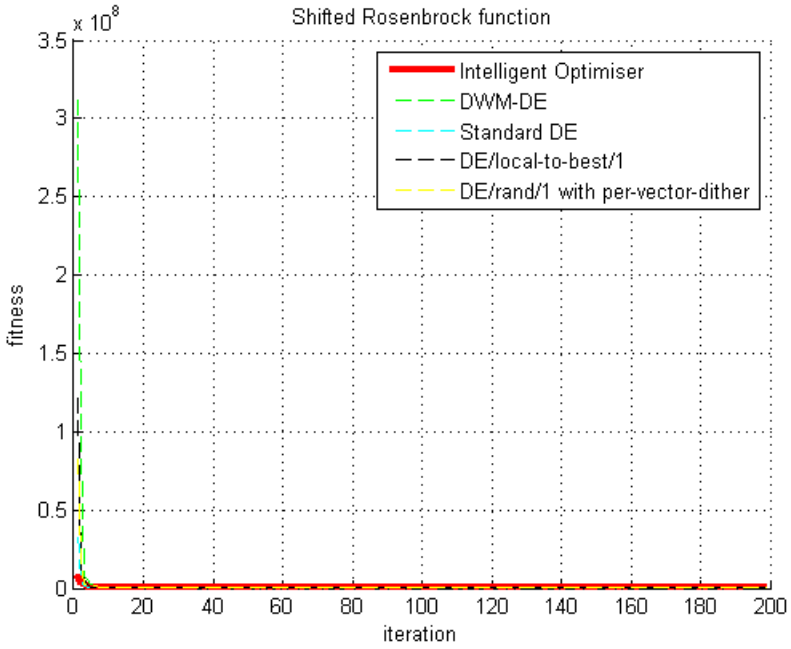
(a)


Shifted Schwefel function
x 10^5
fitness
iteration
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither

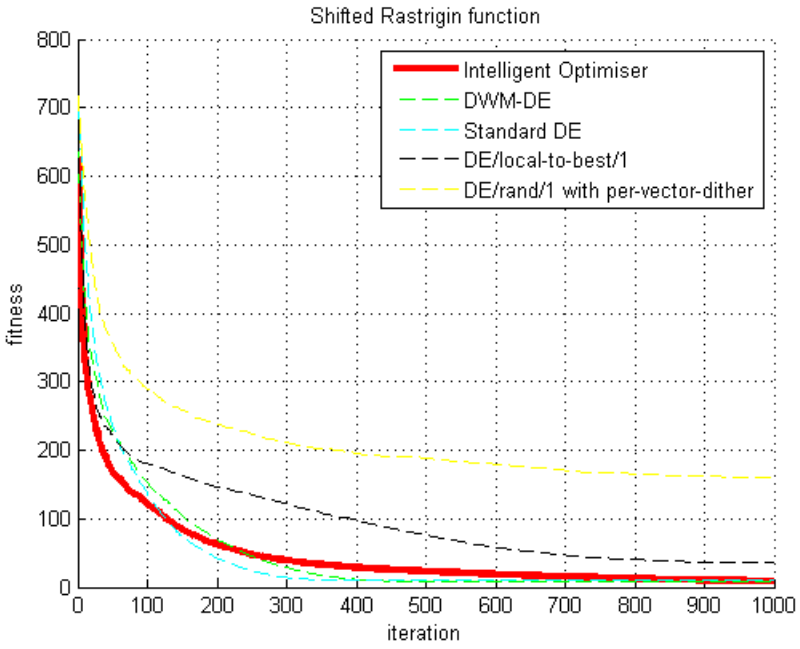
(b)


Shifted rotated Elliptic function
x 10^7
fitness
iteration
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
0
1
2
3
4
5
6
0
20
40
60
80
100
120
140
160
180
200

(c)


Shifted Rosenbrock function
x 10^8
fitness
iteration
Intelligent Optimiser
DWM-DE
Standard DE
DE/local-to-best/1
DE/rand/1 with per-vector-dither
0
0.5
1
1.5
2
2.5
3
3.5
0
20
40
60
80
100
120
140
160
180
200

(d)



(e)

Fig. 4.11. Results for functions with shift and rotate.

IV CONCLUSION

In this chapter, we have proposed an intelligent optimiser that incorporates two identical Wavelet-Mutated Differential Evolution (WM-DE) engines working in parallel on the same fitness function. It employs a fuzzy controller to control the internal parameters of the optimiser based on some expert knowledge. The implementation framework takes advantage of the parallel structure to enhance the optimisation performance. By using the improved algorithm of WM-DE, the proposed intelligent optimiser can achieve a balance between the exploration and exploitation of the solution space for reaching the global solution. A suite of 29 benchmark test functions is employed to test the performance of the proposed intelligent optimiser. In terms of convergence rate and solution quality, the proposed intelligent optimiser could offer better results. Thanks to the parallel structure and the fuzzy controller, better solution reliability can be achieved. Experimental results also show that the intelligent optimiser could offer much better performance when the problem is complex and with a high dimension (>30).

Chapter 5

ECONOMIC LOAD DISPATCH WITH VALVE-POINT LOADING

I BACKGROUND

Economic Load Dispatch (ELD) problem is a fundamental issue when developing a power supply system. Insufficiency and increasing cost of natural resources, and the continuously increasing demand for electric energy have driven engineers to consider the ELD problem for operating modern power systems, where multiple generators are implemented to generate enough total output power to meet the consumer demand. Each generator normally has a unique cost-per-hour characteristic for its operating range. Moreover, each power station should have different costs for fuel and maintenance. ELD is a modelling method to consolidate various factors to formulate a single objective optimisation problem. There are many traditional methods developed to solve the ELD problem. Some examples include the Lagrange multiplier method, Lambda iteration method and Newton-Raphson method. All these methods suffer from the difficulty of obtaining the best result. Moreover, they might take a long computational time

on processing for the solution. A high-quality load dispatching and generation scheduling might not be achieved easily.

In the ELD problem, electric power utilities (companies) are expected to maximise the profit by minimising the operating cost on generating the power to the clients. The load demand and transmission losses must be entertained on providing a stable power supply. For secure operation, the demand of power should be dispatched to different generators correctly, such that the generation capacity limits of individual generators are not exceeded. The major purpose of solving the ELD problem is to control a group of power generators to generate enough electricity with minimum fuel cost, and operates the generators within their physical constraints. The effect of the power generators' valve-point loadings in the fuel cost function and the rate limits of the generators introduce the nonlinear behaviour to the ELD problem. As a result, the objective function for the ELD with Valve-Point Loading (ELD-VPL) problem are multimodal, highly nonlinear, discontinuous in the solution space, high in dimension, and highly constrained. To solve this problem, we have to employ a good algorithm for searching the globally optimal solution.

II PROBLEM FORMULATION

As discussed above, ELD-VPL is to control a group of power generators to generate enough electricity with minimum fuel cost, and operates the generators within their physical constraints. An optimiser is employed that aims at minimising the objective function of the ELD problem. This objective function is defined as follows:

$$\sum_{i=1}^{n} C_i(P_{L_i}) \tag{5.1}$$

where n is the total number of generators in the system. $C_i(P_{Li})$ indicates the i-th generator's operation fuel cost function. The total load demand of the ELD problem is defined as follows.

$$D_L = \sum_{i=1}^{n} P_{L_i} - P_{Loss} \tag{5.2}$$

$$P_{L_{i,min}} \le P_{L_i} \le P_{L_{i,max}}; i = 1,2,\cdots,n \tag{5.3}$$

where the output power of the generator i is denoted as P_{L_i}, the power loss due to transmission is denoted as P_{Loss}, $P_{L_{i,max}}$ is the maximum output power of the i-th generator and $P_{L_{i,min}}$ is the minimum output power of the i-th generator. The operation fuel cost function for each generator is defined as follows.

$$C_i(P_{L_i}) = a_i P_{L_i}^2 + b_i P_{L_i} + c_i \tag{5.4}$$

There are three coefficients in the operation fuel cost function: a_i, b_i, and c_i. If a_i is not equal to zero, the fuel cost function is quadratic.

In the deployment environment of the power system, valve-point effects should be considered for each power generator. The effects of valve points can be modelled with a rectified sinusoidal term and should be added to the operation fuel cost function for each generator. The resulting operation fuel cost function is given by:

$$C_i(P_{L_i}) = a_i P_{L_i}^2 + b_i P_{L_i} + c_i + \left| e_i \times \sin(f_i \times (P_{L_{i,\min}} - P_{L_i})) \right| \quad (5.5)$$

The sinusoidal term introduces two additional coefficients: e_i and f_i. In the ELD-VPL problem, each power generator is constrained by multivalve steam turbines. The multivalve steam turbines introduce a large variation in the fuel cost functions. Fig. 5.1 shows the valve-point effect on the operation fuel cost function of one of the generators. The generator has a_i = 0.00690, b_i = 6.73, c_i = 94.705, e_i = 100, f_i = 0.084, $P_{L_{i,min}}$ = 36 MW and $P_{L_{i,\max}}$ = 114 MW. In the figure, we can see that the valve-point loading introduces many ripples on the fuel-cost curve, which is practically due to the introduced ripples in the heat-rate curves. They increase the difficulty of solving the ELD-VPL problem.

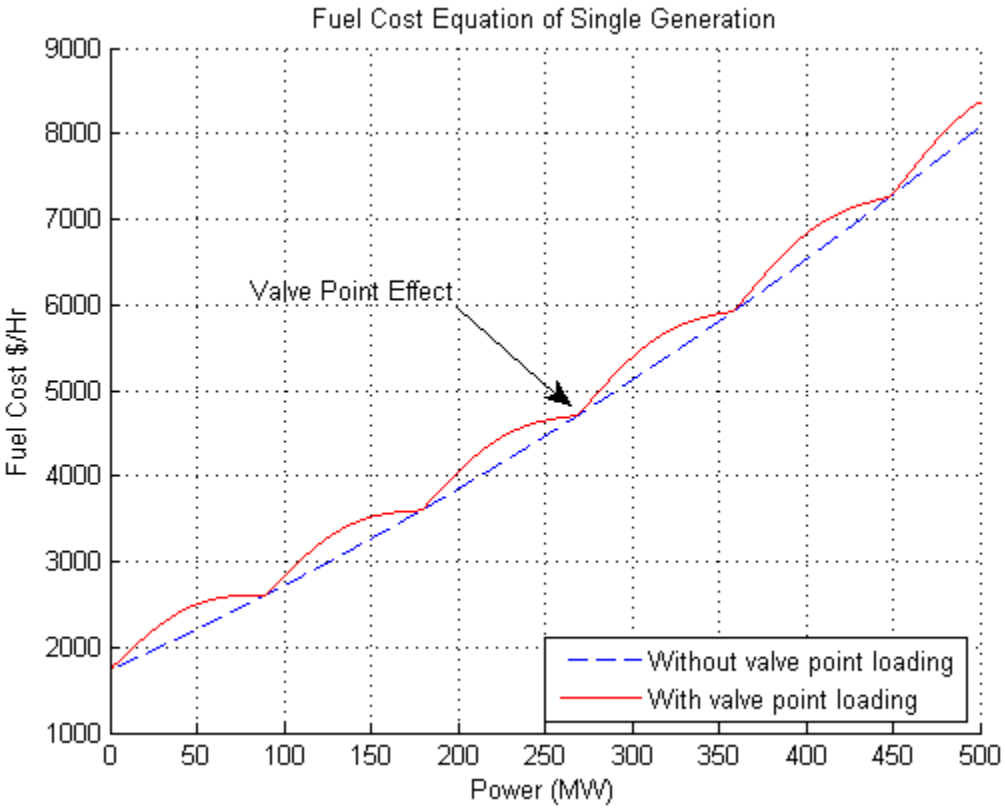


Fig. 5.1. The valve-point effect.

To allow the proposed intelligent optimiser and DWM-DE to determine the minimum cost for the ELD-VPL problem, the power loading of each generator can be formatted as a solution vector. The solution vector can be written as follows.

$$\mathbf{P} = [P_{L_1}\, P_{L_2}\, P_{L_3} \cdots P_{L_{n-1}}] \tag{5.6}$$

$$P_{L_n} = D_L - \sum_{i=1}^{n-1} P_{L_i} + P_{Loss} \tag{5.7}$$

We do not consider the power loss in this thesis. As a result, $P_{Loss} = 0$ and we have

$$P_{L_n} = D_L - \sum_{i=1}^{n-1} P_{L_i} \tag{5.8}$$

The ELD-VPL problem is formulated as an optimisation problem, which is used to minimise the total fuel cost based on (5.5).

III THE EXPERIMENT RESULTS

To test the performance of the proposed DWM-DE algorithm and the intelligent optimiser, two ELD-VPL problems with 13 and 40 generators are considered in this thesis. The results obtained are compared with those reported in the literature. Both the 13- and 40-generator systems have non-convex solution spaces with many local minima. As a result, the global minimum is difficult to determine. For the 13-generator system, the total load demand of 1800MW is tested. For the 40-generator system, the total load demand of 10500MW is tested.

Table 5.1. Parameters for the 13-Generator System.

Unit(i)	a_i	b_i	c_i	e_i	f_i	$P_{L_{i,min}}$	$P_{L_n,\min}$
1	0.00028	8.10	550	300	0.035	0	680
2	0.00056	8.10	309	200	0.042	0	360

3	0.00056	8.10	307	150	0.042	0	360
4	0.00324	7.74	240	150	0.063	60	180
5	0.00324	7.74	240	150	0.063	60	180
6	0.00324	7.74	240	150	0.063	60	180
7	0.00324	7.74	240	150	0.063	60	180
8	0.00324	7.74	240	150	0.063	60	180
9	0.00324	7.74	240	150	0.063	60	180
10	0.00284	8.60	126	100	0.084	40	120
11	0.00284	8.60	126	100	0.084	40	120
12	0.00284	8.60	126	100	0.084	55	120
13	0.00284	8.60	126	100	0.084	55	120

Table 5.2. Parameters for the 40-Generator System.

Unit(i)	a_i	b_i	c_i	e_i	f_i	$P_{L_{i,min}}$	$P_{L_n,\min}$
1	0.00690	6.73	94.705	100	0.084	36	114
2	0.00690	6.73	94.705	100	0.084	36	114
3	0.02028	7.07	309.54	100	0.084	60	120
4	0.00942	8.18	369.03	150	0.063	80	190
5	0.01140	5.35	148.89	120	0.077	47	97
6	0.01142	8.05	222.33	100	0.084	68	140
7	0.00357	8.03	278.71	200	0.042	110	300
8	0.00492	6.99	391.98	200	0.042	135	300
9	0.00573	6.60	455.76	200	0.042	135	300
10	0.00605	12.9	722.82	200	0.042	130	300
11	0.00515	12.9	635.20	200	0.042	94	375
12	0.00569	12.8	654.69	200	0.042	94	375
13	0.00421	12.5	913.40	300	0.035	125	500

14	0.00752	8.84	1760.40	300	0.035	125	500
15	0.00708	9.15	1728.30	300	0.035	125	500
16	0.00708	9.15	1728.30	300	0.035	125	500
17	0.00313	7.97	647.85	300	0.035	220	500
18	0.00313	7.95	649.69	300	0.035	220	500
19	0.00313	7.97	647.83	300	0.035	242	550
20	0.00313	7.97	647.81	300	0.035	242	550
21	0.00298	6.63	785.96	300	0.035	254	550
22	0.00298	6.63	785.96	300	0.035	254	550
23	0.00284	6.66	794.53	300	0.035	254	550
24	0.00284	6.66	794.53	300	0.035	254	550
25	0.00277	7.10	801.32	300	0.035	254	550
26	0.00277	7.10	801.32	300	0.035	254	550
27	0.52124	3.33	1055.10	120	0.077	10	150
28	0.52124	3.33	1055.10	120	0.077	10	150
29	0.52124	3.33	1055.10	120	0.077	10	150
30	0.01140	5.35	148.89	120	0.077	47	97
31	0.00160	6.43	222.92	150	0.063	60	190
32	0.00160	6.43	222.92	150	0.063	60	190
33	0.00160	6.43	222.92	150	0.063	60	190
34	0.00010	8.95	107.87	200	0.042	90	200
35	0.00010	8.62	116.58	200	0.042	90	200
36	0.00010	8.62	116.58	200	0.042	90	200
37	0.01610	5.88	307.45	80	0.098	25	110
38	0.01610	5.88	307.45	80	0.098	25	110
39	0.01610	5.88	307.45	80	0.098	25	110
40	0.00313	7.97	647.83	300	0.035	242	550

The parameters for the 13- and 40-generator systems are shown in Tables 5.1 and 5.2 respectively. On using the DWM-DE algorithm, the following experiment settings are used:

- Shape parameter of the wavelet mutation: 1. (This value is not very critical. See the discussion in Chapter 3.)
- Parameter λ for the monotonic increasing function: 10000. (This value is not very critical. See the discussion in Chapter 3)
- Initial population: It is generated uniformly at random.
- Numbers of iteration: 1000.
- Number of population vectors: 50.
- Crossover probability constant: 0.5.

On using the proposed intelligent optimiser, the following experiment settings are used:

- Shape parameter of the wavelet mutation: 1 (This value is not very critical. See the discussion in Chapter 3)
- Parameter λ for the monotonic increasing function: 10000. (This value is not very critical. See the discussion in Chapter 3)
- Initial population: It is generated uniformly at random.
- Numbers of iteration: 1000.
- Number of population vectors: 25 x 2.

All results shown in this chapter are averaged ones out of 100 trials. The experiment results in terms of mean cost value, best cost value, and standard deviation are presented in Table 5.3.

Table 5.3. Result of the ELD-VPL Problem.

Number of Generators	Load		Intelligent Optimiser	DWM-DE	standard DE	DE/local-to-best/1	DE/rand/1 with per-vector-dither
13	1800MW	Mean	**17992.02**	17996.43	18185.27	18078.82	18213.61
		Best	**17978.23**	**17972.78**	18104.61	17982.91	18077.96
		Std Dev	**11.41**	20.85	51.61	47.95	45.34
40	10500 MW	Mean	**121478.8**	121521.79	121834.62	123363.29	122490.90
		Best	**121420.01**	121431.63	121530.99	121971.29	122188.14
		Std Dev	**36.93**	53.27	172.74	610.10	89.64

Table 5.4. Comparison with Other Published Results for the 13-Generator System (Load = 1800MW).

Intelligent Optimiser	DWM-DE	DEC-SQP	IGA
17978.23	17972.78	**17938.95**	18069.40

Table 5.5. Comparison with Other Published Results for the 40-Generator System (Load = 10500MW).

Intelligent Optimiser	DWM-DE	MPSO	DEC-SQP	DE/BBO	QPSO	NPSO-LRS	SOH-PSO
121420.01	**121431.63**	122252.26	121741.97	121426.95	121448.21	121664.43	121501.14

From the results obtained in the experiments, we find that the proposed intelligent optimiser and the DWM-DE algorithm perform much better than the other DE methods. The average cost offered by the intelligent optimiser for the 13-generator system is $17992.02, and the best result (minimum cost) offered by DWM-DE is $17972.78. The proposed intelligent optimiser provides the lowest standard deviation. For the 40-generator system, the intelligent optimiser can provide the best performance. The average cost is $121478.8, and the best result (minimum cost) is $121420.01. The two proposed method could offer good results in terms of standard deviation. Thanks to the wavelet operations, the DWE-DE algorithm searches the solution space more effectively. In the proposed intelligent optimiser, the adaptive updating of the parameter values and the population analysis using the Student T-Test bring proper adjustment to the parameters of the two DE engines according to the populations' behaviour. As a result, better performance can be achieved. The solution reliability is important for the ELD-VPL problem because a reliable optimisation method can offer better quality of the power generation service. The two proposed methods could offer better performance than the conventional methods in terms of solution quality, convergence speed and solution reliability. Both methods could be applied to solve the ELD-VPL problem successfully.

Table 5.4 summarises the best results obtained by the intelligent optimiser, DWM-DE, DEC-SQP [Coelho 06] and IGA [Chen 95] for comparison. The results show that the DEC-SQP performs the best for the 13-generator system. The best cost is $17938.95, while the best cost of DWM-DE is $17972.78 and the best cost of the intelligent optimiser is $179782.23. Although the DWM-DE algorithm and the intelligent optimiser cannot offer the best result, the result is already very near to the DEC-SQP method. As the dimension of the 13-generator system is

relatively low, the wavelet based mutations and the adaptive parameters tuning might not be able to enhance the searching process very effectively.

For the 40-generator system, the best results of the intelligent optimiser, DWM-DE, MPSO [Victoire 04], DEC-SQP [Coelho 06], DE/BBO [Bhattacharya 10], QPSO[Meng 10], SOH-PSO[Chaturvedi 08] and NPSO-LRS [Victoire 04] are summarised in Table 5.5. Among all the methods, our proposed methods offer the best results. For the 40-generator system case, the best cost (minimum cost) is $121,420.01 offered by the intelligent optimiser. Since the dimension of the 40-generator system is large, the wavelet-based mutations can enhance the searching process effectively and reduce the chance of trapping in some local minimum. For the ELD-VPL problem, to obtain the best result, we suggest applying the proposed methods for systems of high dimensions; for example, a dimension higher than 30.

IV CONCLUSION

In this chapter, the proposed intelligent optimiser and DWM-DE algorithm are employed to determine the minimum operation cost for the Economic Load Dispatch with Valve-Point Loading (ELD-VPL) problem. The ELD-VPL problem is multimodal, discontinuous, highly nonlinear, of high dimension, and highly constrained. Due to the nature of the Valve-Point Loading (VPL) effect, many ripples are introduced on the fuel-cost curve, which further increase the difficulty of the problem. Two different requirements of the ELD-VPL problem have been tested. It is observed that the two proposed methods give satisfactory optimal costs when compared with other techniques in the literature. Moreover, the experiment results show that the two proposed methods could offer better performance than the conventional methods in terms of

solution quality, convergence speed and solution reliability. Thanks to the wavelet operations, the DWE-DE algorithm searches the solution space more effectively. Thanks to the fuzzy controller and the T-Test analysis in the proposed intelligent optimiser, the population information can be captured to change the parameter values of the optimiser adaptively in order to obtain better searching performance.

Chapter 6

HYPOGLYCAEMIA DETECTION USING FUZZY INFERENCE SYSTEM

I BACKGROUND

Low level of blood glucose is an important issue for the human body. It may happen owing to high energy-consumption, action of insulin and food ingestion. When the human body suffer from a low level of blood glucose, hypoglycaemia occurs. Non-diabetic persons are not common to have hypoglycaemia [Yale 04]. The problem may be due to long-term starvation, superfluous insulin, innate problem, drugs, alcohol, insufficient hormone generation, and organ defect [DCCT 95]. Study reported that the diabetic patients have higher chance to develop hypoglycaemia in their body if they have been treated with insulin. Another study reported that young patient with intensive glycaemia control will have high frequency of developing hypoglycaemia [Pickup 00]. Different people may have different level of blood glucose to

develop hypoglycaemia in their body. In general, if the body maintains above 70 mg/dL (3.9 mmol/L) for fasting glucose, it is considered as healthy for adults. If the blood glucose level drops below 55 mg/dL, the body may start to develop hypoglycaemia [DCCT 95]. If the blood glucose level is below 50 mg/dL (2.8 mmol/L), the body is suffering from hypoglycaemia. Medical treatment is required for the patients through, for example, injection or infusion of glucagon. Maintaining a certain level of glucose is necessary for the nervous system and brain to function property. When the central nervous systems detect the presence of hypoglycaemia, it will automatically reduce the cerebral glucose consumption (neuroglycopenic symptoms) [Maia 07]. Some symptoms can be activated before neuroglycopenic symptoms occur, for example, sweating, weakness, fatigue, blurry or double vision, weakness extreme hunger and headache. As a result, the patient can be aware of the presence of hypoglycaemia. If the human brain does not have enough supply of glucose, the body may suffer neuroglycopenic symptoms, for example, loss of consciousness (coma), seizures, and confusion [Amir 07].

It is particularly dangerous if hypoglycaemia happens at night. It is because the patient may not be able to take immediate response. As a result, mild episodes of hypoglycaemia may become serious. Study reported that more that 50% of all serious episodes of hypoglycaemia happened at evening. Even with the modest insulin elevation, serious hypoglycaemia may result owing to deficient glucose counter-regulation. Moreover, the dawn phenomenon makes the handling of hypoglycaemia more difficult [Weinstein 07]. The dawn phenomenon shows that the demand for insulin by the human body decreases between midnight and 5 am. Between 5 am and 8 am, the demand for insulin by the human body increases. As a result, the technology of detecting presence of hypoglycaemia is very challenging in the medical industry [Caduff 09] [Cho 08] [Chu 08].

In this study, a real-time detector is constructed to detect the presence of hypoglycaemia. The detector is realised by a fuzzy inference system (FIS). The relationship between physiological signals and the presence of hypoglycaemia are captured by the FIS to perform the detection. Four physiological signals are employed as the input of the detector. They are the heart rate (HR), the change of HR with time (ΔHR), the corrected QT interval (QT_c) of the electrocardiogram (ECG) signal, and the change of QT_c with time (ΔQT_c). The output of the detector is the presence of hypoglycaemia (*h*). The output of the FIS is a binary value of true or false. To realise the classification, the FIS is required to construct the relationship between the physiological inputs to the presence of hypoglycaemia by using fuzzy logic. The FIS contains four major elements. They are fuzzy-rule base, inference engine, fuzzification, and defuzzification. Linguistic variables are used to represent the physiological data (HR, QT_c, ΔHR and ΔQT_c) such that they can be processed by fuzzy logic to perform decision-making. In the inference engine of the FIS, a number of fuzzy if-then rules are used to support reasoning for doing classification.

Training process is required for the fuzzy inference system (FIS) to capture the relationship between the physiological signals and the presence of hypoglycaemia for patients with type-1 diabetes mellitus (T1DM) patients. In this thesis, supervised learning is adopted as the training method. The dataset with well-defined class labels are input to the FIS during the training process. Optimisation can be employed in the training to determine the best parameter values for constructing the fuzzy rules and fuzzy membership functions inside the FIS. The proposed intelligent optimiser discussed in Chapter 4 and the DWM-DE algorithm discussed in Chapter 3 are employed to determine the best parameter values for the proposed FIS to perform the classification.

In this study, the sensitivity (ξ_{sen}) and specificity (η_{spec}) are used to evaluate the performance of the classification. If the classification can identify all the sick people, the sensitivity is equal to 100%. If the classification can identify all the healthy people, the specificity is equal to 100%. To offer a reliable detection of the presence of hypoglycaemia, high values of sensitivity and specificity must be obtained in both the training and testing. For clinical requirements of classification, it is suggested that the value of sensitivity must be large than 70% and the value of specificity must be large than 50%. To realise the clinical requirements, a multi-objective optimisation is adopted.

In the training process, with the use of training set only, it may introduce a phenomenon called overtraining [Chan 11a] [Chan 11b]. The trained system performance may be degraded if overtraining occurs. Overtraining means that the trained system bias on the training set data and lose the ability to handle general data set. To minimise the effect of overtraining, we proposed to introduce another data set to the training process. This data set is called the validation set. As a result, three data sets are required for the proposed FIS.

II PROBLEM FORMULATION

In this thesis, an FIS is employed to detect the presence of hypoglycaemia for type-1 diabetes mellitus (T1DM) patients. The proposed Double Wavelet Mutated Differential Evolution (DWM-DE) (discussed in Chapter 3) and the intelligent optimiser with two Wavelet-Mutated Differential Evolution (WM-DE) engines (discussed in Chapter 4) are employed to tune the internal parameters of the FIS. Fig. 6.1 demonstrates how the FIS works. The proposed FIS

consists of four inputs and one output. The inputs are the heart rate (HR), the corrected QT interval of the electrocardiogram signal (QT_c), the change of heart rate (ΔHR), and the change of QT_c (ΔQT_c). The output is the presence of hypoglycaemia (*h*), which is a binary value of true or false. To realise the classification, the FIS is required to construct the relationship between the physiological inputs and the binary output by using fuzzy logic.

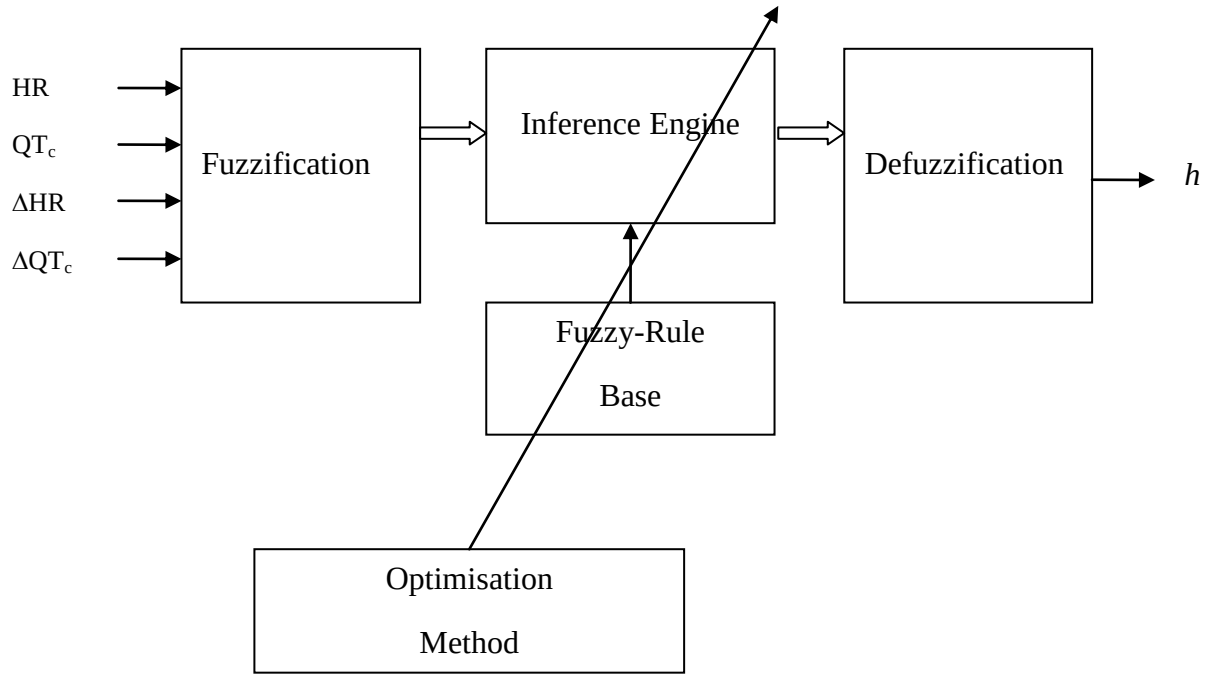


Fig. 6.1. Fuzzy inference system (FIS).

The ECG signal governs two of the physiological inputs of the FIS. Fig. 6.2 shows an example two-cycle ECG waveform. The ECG signal being investigated involves the parameters in the depolarisation and repolarisation stages of electrocardiography. An ECG signal contains 4 important points. They are the T wave peak (T_p), R peak, Q points, and the T wave end (T_e). In this study, we measure the interval between the Q point and the T wave peak (T_p). This interval

is called the QT interval. The interval between two R peaks within the two-cycle ECG signal is also measured. This interval is called the RR interval. With the QT interval and RR interval, we could generate the physiological signal QT_c by dividing RR with QT. In addition, the heart rate is determine by RR divided by 60.

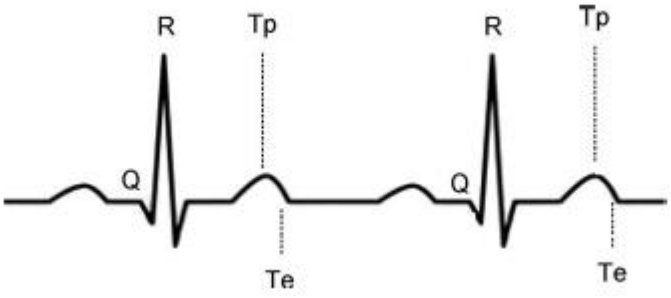


Fig. 6.2. ECG signal.

The FIS contains four major elements: fuzzy-rule base, inference engine, fuzzification, and defuzzification. Linguistic variables are used to represent the physiological data (HR, QT_c, ΔHR and ΔQT_c) such that they can be processed by fuzzy logic to make decision. The physiological data are mapped to the designed membership functions during fuzzification. The fuzzy membership function is defined as follows:

$$f_m(x(t);\sigma_m,c_m) = e^{\frac{-(x-c_m)^2}{2\sigma_m^2}} \quad (6.1)$$

where $m=1, 2,\ldots,m_f$. It is a bell-shaped fuzzy membership function where $x(t)$ is the physiological input. The number of membership functions is denoted as m_f. The standard deviation of the bell-shaped fuzzy membership function is denoted as σ_m and the mean value of the membership function is denoted as c_m. Fig. 6.3 shows a set of bell-shaped fuzzy membership functions for an input variable. It contains three linguistic terms (M1, M2, and M3). The input range is between 0 and 1.

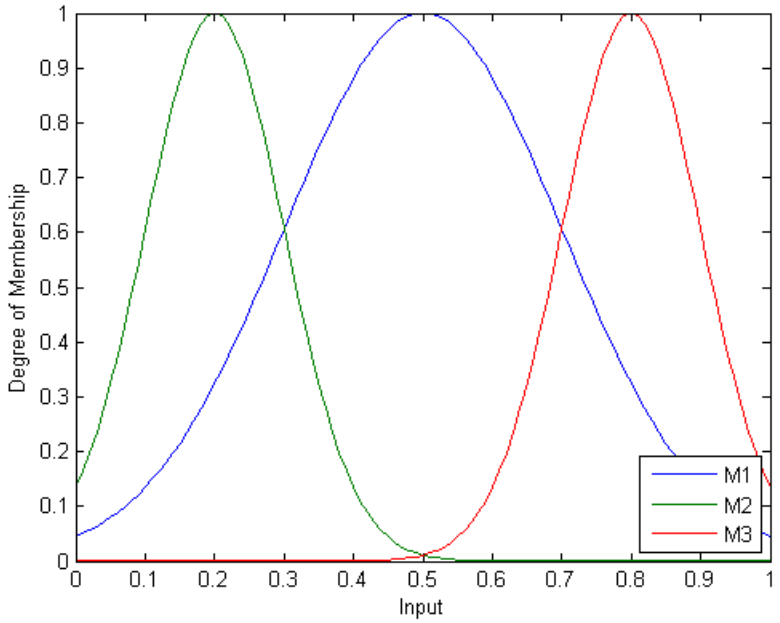


Fig. 6.3. Bell-shaped fuzzy membership functions.

In the proposed FIS, the bell-shaped fuzzy membership functions for the four physiological inputs are defined as follows:

$$f_{HR,m}(HR(t), \sigma_{HR,m}, c_{HR,m}) = e^{\frac{-(HR(t)-c_{HR,m})^2}{2(\sigma_{HR,m})^2}} \quad (6.2)$$

$$f_{QTc,m}(QTc(t), \sigma_{QTc,m}, c_{QTc,m}) = e^{\frac{-(QTc(t)-c_{QTc,m})^2}{2(\sigma_{QTc,m})^2}} \quad (6.3)$$

$$f_{\Delta HR,m}(\Delta HR(t), \sigma_{\Delta HR,m}, c_{\Delta HR,m}) = e^{\frac{-(\Delta HR(t)-c_{\Delta HR,m})^2}{2(\sigma_{\Delta HR,m})^2}} \quad (6.4)$$

$$f_{\Delta QTc,m}(\Delta QTc(t), \sigma_{\Delta QTc,m}, c_{\Delta QTc,m}) = e^{\frac{-(\Delta QTc(t)-c_{\Delta QTc,m})^2}{2(\sigma_{\Delta QTc,m})^2}} \quad (6.5)$$

In the inference engine of the FIS, a number of fuzzy rules are used to perform reasoning for doing classification. The rules used in the proposed FIS are the fuzzy if-then rules. The format of the rules are defined as follows:

Rule τ:	**IF**	$HR(t)$	is	$f_{HR,m}(HR(t), \sigma_{HR,m}, c_{HR,m})$
	AND	$QTc(t)$	is	$f_{QTc,m}(QTc(t), \sigma_{QTc,m}, c_{QTc,m})$
	AND	$\Delta HR(t)$	is	$f_{\Delta HR,m}(\Delta HR(t), \sigma_{\Delta HR,m}, c_{\Delta HR,m})$
	AND	$\Delta QTc(t)$	is	$f_{\Delta QTc,m}(\Delta QTc(t), \sigma_{\Delta QTc,m}, c_{\Delta QTc,m})$,
	THEN	$y(t)$	is	w_τ.

where τ is the rule number between 1 and the number of rules in the fuzzy rule base (n_r). (6.6) shows how to determine the value of n_r with the number of inputs (n) of the FIS.

$$n_r = (m_f)^n \quad (6.6)$$

After the fuzzification of the inputs, aggregation of each rule output is performed to generate an output as a single fuzzy set. The output of aggregation for the corresponding fuzzy rule is defined as follows:

$$o_\tau = f_{HR,m}(HR(t), \sigma_{HR,m}, c_{HR,m}) \times f_{QT_c,m}(QT_c(t), \sigma_{QT_c,m}, c_{QT_c,m}) \times f_{\Delta HR,m}(\Delta HR(t), \sigma_{\Delta HR,m}, c_{\Delta HR,m}) \times f_{\Delta QT_c,m}(\Delta QT_c(t), \sigma_{\Delta QT_c,m}, c_{\Delta QT_c,m}) \quad (6.7)$$

After the inference process, defuzzification is required to transform the fuzzy output to some crisp output. In the proposed FIS, the process of defuzzification is defined as follows:

$$y(t) = \frac{\sum_{\tau=1}^{n_r} o_\tau w_\tau}{\sum_{\tau=1}^{n_r} o_\tau} \quad (6.8)$$

where $w_\tau \in [-1,1]$, $\tau = 1, 2, \ldots n_r$ are fuzzy singletons to be determined in the training process. The actual output of the FIS is realised as:

$$h(t) = \begin{cases} +1, & if\ y(t) \geq 0. \\ -1, & otherwire \end{cases} \quad (6.9)$$

As a result, the output of the classifier is binary. If the output is positive, it indicates the presence of hypoglycaemia.

In the proposed system, the proposed intelligent optimiser and DWM-DE are employed to determine the best parameters for the fuzzy rules and membership functions in the FIS. The detailed implementations of the intelligent optimiser and DWM-DE have been discussed in Chapter 4 and Chapter 3 respectively. In this study, the sensitivity (ξ_{sen}) and specificity (η_{spec}) are used to evaluate the performance of the classification. The definitions of sensitivity and specificity are given as follows.

$$\xi_{sen} = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{6.10}$$

$$\eta_{spec} = \frac{N_{TN}}{N_{TN} + N_{FP}} \tag{6.11}$$

N_{TP} means the number patient with illness identified correctly; N_{FN} means the number of patient with illness not correctly identified; N_{FP} means the number of healthy people incorrectly identified as sick; N_{TN} means the number of healthy people correctly identified [Freedman 05] [Altman 94]. The sensitivity (ξ_{sen}) and specificity (η_{spec}) have the maximum value of one and minimum value of zero. If the classification can identify all the sick people, the sensitivity is equal to one. If the classification can identify all the healthy people, the specificity is equal to

one. To offer a reliable detection of the presence of hypoglycaemia, high values of sensitivity and specificity should be obtained in both the training and testing.

The FIS perform learning by capturing the input-output relationship of the system using some well-defined dataset. Two data sets are often required: a training set and a testing set. The training set is used for the FIS to learn the input-output relationship of the system. The testing set is used to evaluate the performance of the trained system. However, by using two sets of data only, it may introduce a phenomenon called overtraining. The trained system's performance may be degraded if overtraining occurs, which means that the trained system is biased by the training set data. The system then loses the ability to handle general data set. To minimise the effect of overtraining, we propose to introduce another data set to the training process. This data set is called the validation set, which is also used in the training process. As a result, three data sets are required for the proposed FIS.

For the training process, fitness functions are required for the optimiser to evaluate the performance of the trained system. To implement the training process with validation, four fitness functions are introduced in training process for the FIS. The definition of the four fitness functions are given as follows:

$$f_1 = \begin{cases} \xi_{target} & if\ \ \xi_{train} \geq \xi_{target} \\ \xi_{train} & otherwise \end{cases} \tag{6.12}$$

$$f_2 = \begin{cases} \eta_{target} & if\ \ \eta_{train} \geq \eta_{target} \\ \eta_{train} & otherwise \end{cases} \tag{6.13}$$

$$f_3 = \begin{cases} \xi_{target} & if \ \xi_{val} \geq \xi_{target} \\ \xi_{val} & otherwise \end{cases} \tag{6.14}$$

$$f_4 = \begin{cases} \eta_{target} & if \ \eta_{val} \geq \eta_{target} \\ \eta_{val} & otherwise \end{cases} \tag{6.15}$$

The sensitivity offered by the training set and the validation set are denoted as ξ_{train} and ξ_{val} respectively. The specificity offered by the training set and the validation set are denoted as η_{train} and η_{val} respectively. The target value for the sensitivity and specificity are denoted as ξ_{target} and η_{target} respectively. The values for the target sensitivity and specificity are not constant during the optimisation (training) process. Fig. 6.4 shows the pseudo code to demonstrate the process of updating the target value for sensitivity and specificity.

```
begin
Initialise the target value
ξtarget =0.10
ηtarget= 0.10
while (not termination condition) do
    begin
      Output (ξtrain,ξval) by Equation (6.10)
      Output (ηtrain,ηval) by Equation (6.11)
      Output (f1) by Equation (6.12)
      Output (f2) by Equation (6.13)
      Output (f3) by Equation (6.14)
      Output (f4) by Equation (6.15)
      If (f1=ξtarget& f3=ξtarget)
      then ξtarget=ξtarget+ 0.01
      if (f2=ηtarget& f4=ηtarget)
      then ηtarget =ηtarget + 0.01
     end
end
```

Fig. 6.4. Pseudo code for the training and validation.

The target values for the specificity and sensitivity are set to be very small at the initialization. In this thesis, we set the target values for specificity and sensitivity to be 0.10 for the initialization. During the optimisation, the target value for sensitivity (ξ_{target}) increases if and only if the sensitivities offered by the training set (ξ_{train}) and validation set (ξ_{val}) are both equal to the target value for sensitivity (ξ_{target}). Similar to the sensitivity, the target value for specificity (η_{target}) increases if and only if the specificities offered by the training set (η_{train}) and validation set (η_{val}) are both equal to the target value for specificity (η_{target}). The optimisation process will keep increasing the target values for ξ_{target} and η_{target} until the end of optimisation. This mechanism can offer a balance between the training set and validation set when performing the training process. For example, if the target sensitivity is equal to 0.70, the sensitivity of training is equal to 0.76, and the sensitivity of validation is equal to 0.90, then the fitness value of f_1 and f_3 will be set at the same value of the target sensitivity. Since both the sensitivities offered by the training set and validation set are equal to the target sensitivity, the target sensitivity will be increased. The validation set offer much higher value than the one offered by the training set, but the optimiser considers them as the same. As a result, we could minimise the effect of overtraining by avoiding the training process from biasing to the training set or validation set.

To realise the training discussed above that involves four fitness functions, a multi-objective optimisation is done. Pareto optimisation is one of the common methods to handle multi-objective optimisation problems. The major objective of the Pareto optimisation is to optimise multi-objective functions at the same time, and every objective function has equal weighting. It means that every objective has the same importance. The basic operation principle of Pareto optimisation is that if the new trial solution declines any objective function in the system, it is a poor solution. If the trial solution can improve more than one objective function

and do not decline any objective function in the system, it is a good solution and will be accepted. As a result, the Pareto optimisation will not be dominated by any objective function in the system. The idea of Pareto optimisation can be realised in the selection process of the intelligent optimiser and DWM-DE. The selection process of multi-objective optimisation is defined as follows:

$$\mathbf{x}_{i,g+1} = \begin{cases} \bar{\mathbf{u}}_{i,g} & \text{if } \forall i \ f_i(\bar{\mathbf{u}}_{i,g}) \le f_i(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \tag{6.16}$$

The i-th objective function is denotes as $f_i(\cdot)$. Fig. 6.5 demonstrates the operation of the multi-objective DWM-DE. Fig. 6.6 demonstrates the operation of the multi-objective WM-DE inside the intelligent optimiser.

```
begin
Initialise the population
    while (not termination condition) do
    begin
      Update the new value of F by equation (3.12)
      Mutation operation by equation (3.2)
      Crossover operation by equation (3.3)
      Modifying the trail population vectors based on equation (3.15)
      Evaluation of the fitness functions
      Select the best vector by equation (6.16)
    end
end
```

Fig. 6.5. Pseudo code for multi-objective DWM-DE.

```
begin
Initialise the population
    while (not termination condition) do
    begin
      Mutation operation by equation (3.2)
      Crossover operation by equation (3.3)
      Modifying the trail population vectors based on equation (3.15)
      Evaluation of the fitness functions
      Select the best vector by equation (6.16)
    end
end
```

Fig. 6.6. Pseudo code for multi-objective WM-DE.

III EXPERIMENT RESULTS

Fifteen children with TIDM at the Princess Hospital for Children in Perth, Western Australia, Australia are invited to join the study of hypoglycaemia. Their average age is around fifteen years old. They are required to measure their body signals and blood glucose levels for ten hours at midnight. This study is approved by Woman's and Children's Health Service, Department of Health, Government of Western Australia, and with informed consent.

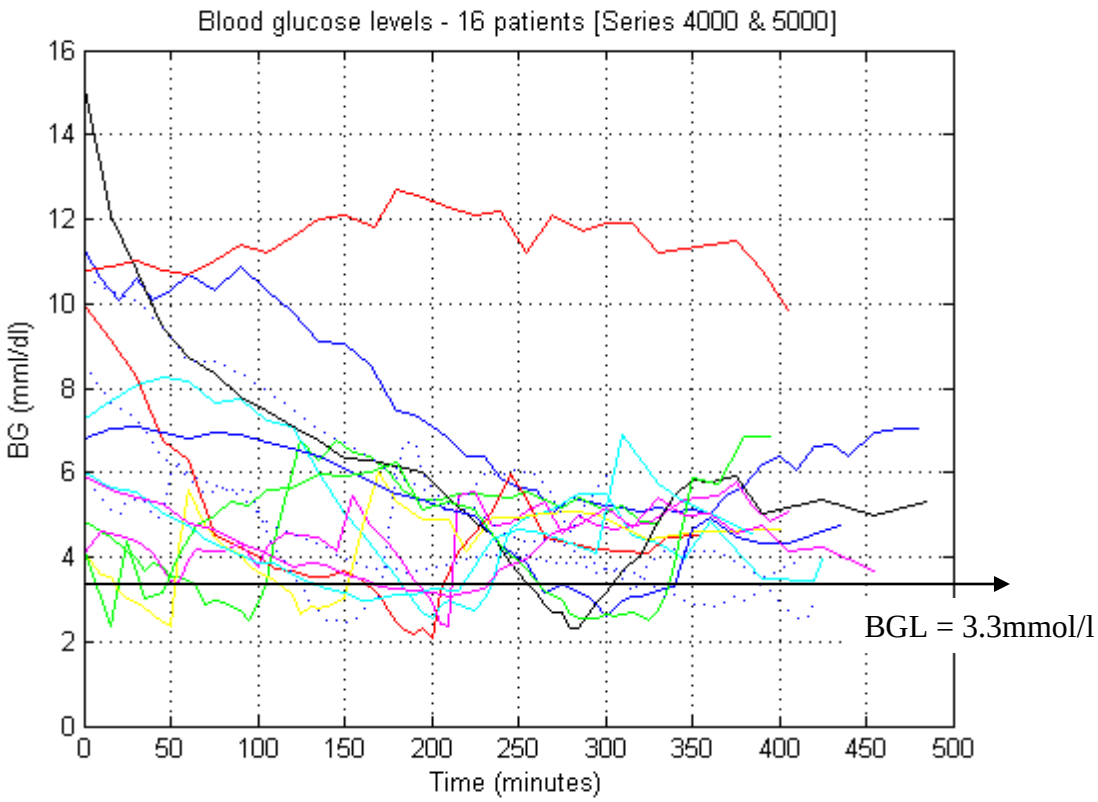


Fig. 6.7. Actual BG-Level profiles of 16 T1DM children.

Fig. 6.5 shows the blood glucose levels (BGL) for the fifteen children with TIDM. Each patient is required to measure his or her blood glucose for around 450 minutes. All the patients have significant changes of their blood glucose levels with the presence of hypoglycaemia. The whole dataset contains samples of both presence and absence of hypoglycaemia data. In this study, the body signals collected are normalised to reduce the effect of individual variability of patients.

The obtained clinical dataset is used to train the fuzzy inference system (FIS) to perform the detection of the presence of hypoglycaemia. In this thesis, the presence of hypoglycaemia is defined as a blood glucose level below 3.3mmol/l. Table 6.1 shows the usage of the obtained

clinical dataset. The dataset is divided into three subsets. Each subset has the samples of 5 patients. Each subset has different number of samples depending on which patient is chosen. For the training set, there are 199 samples. For the validation set, there are 177 samples. For the testing set, there are 193 samples.

Table 6.1. Dataset for the Experiments.

Purpose	**Number of Patients**	**Number of data points**
Training	5	199
Validation	5	177
Testing	5	193

The training and testing performance are evaluated by using sensitivity and specificity. In this thesis, 10 different approaches for the hypoglycaemia detection [Sebar 03] [Wang 06] [Ling 11] are used for comparison. They are:

i) The proposed FIS with four inputs (HR, QT_c, ΔHR and ΔQT_c) tuned by the proposed *intelligent optimiser* with the proposed training and validation method.

ii) An evolved fuzzy inference system with two inputs (HR and QT_c) tuned by the proposed *intelligent optimiser* with the proposed training and validation method.

iii) The proposed FIS with four inputs (HR, QT_c, ΔHR and ΔQT_c) tuned by the proposed *DWM-DE* with the proposed training and validation method.

iv) An evolved fuzzy inference system with two inputs (HR and QT_c) tuned by the proposed *DWM-DE* with the proposed training and validation method.

v) A fuzzy inference system with four inputs (HR, QT_c, ΔHR and ΔQT_c) without validation (FIS-4- w/o-v) [Ling 11].

vi) A fuzzy inference system with two inputs (HR and QT_c) without validation (FIS-2- w/o-v) [Ling 11].

vii) A linear multiple regression with 4 inputs (HR, QT_c, ΔHR and ΔQT_c) (LR-4) [Ling 11].

viii) A linear multiple regression with 2 inputs (HR and QT_c) (LR-2) [Ling 11].

ix) An evolved multiple regressions with two inputs (EMR2) [Ling 11].

x) A feed-forward neural network (FFNN) [Ling 11].

The proposed intelligent optimiser with two WM-DE engines is used to optimise the fuzzy rules and membership functions of the FIS in case i) and ii). The configuration of the experiments is given as follows.

- Shape parameter of the wavelet mutation (ζ_{wm}): 1. (This value is not very critical. See the discussion in Chapter 3.)
- Parameter λ for the monotonic increasing function: 10000. (This value is not very critical. See the discussion in Chapter 3.)
- Initial population: It is generated uniformly at random.
- Number of iteration: 5000.
- Population size: 50×2.

On using the DWM-DE algorithm for iii) and iv), the settings of the parameter values are given as follows.

- Shape parameter of the wavelet mutation: 1. (This value is not very critical. See the discussion in Chapter 3.)

- Parameter λ for the monotonic increasing function: 10000. (This value is not very critical. See the discussion in Chapter 3)
- Initial population: Generated randomly and uniformly.
- Number of iteration: 5000.
- Population size: 100.
- Crossover Rate: 0.5.

In the experiment, two configurations of FIS is embedded for testing. The first one is the FIS with the inputs of HR and QT_c (two-input FIS). The second one is the FIS with the inputs of HR, QT_c, ΔHR and ΔQT_c (four-input FIS). The two-input FIS and four-input FIS are tested with different numbers of membership functions (m_f). Table 6.2 and Table 6.3 show the numbers of membership functions involved and numbers of rules involved. The sensitivity and specificity obtained are also reported in these tables. The result for the training dataset, validation dataset and testing dataset are reported separately in the tables. In this study, the experiment for each case has run for 50 trials. The best results obtained by the DWM-DE, the intelligent optimiser, and the other methods are given in Table 6.4.

Table 6.2. Results of using DWM-DE (Average of 50 Trials).

No of	m_f	t_d	Testing	Training	Validation

Inputs			ξ	η	ξ	η	ξ	η
2	3	21	71.07%	40.92%	80.35%	40.84%	83.11%	42.14%
	5	45	73.22%	40.13%	80.56%	42.21%	81.22%	41.24%
	8	96	73.07%	40.21%	81.21%	43.32%	80.00%	42.23%
4	3	105	72.06%	42.22%	82.34%	41.00%	86.24%	44.92%
	5	665	**73.92%**	**51.22%**	**83.33%**	**41.00%**	**90.10%**	**50.95%**

Table 6.3. Results of using Intelligent Optimiser (Average of 50 trials).

No of Inputs	m_f	t_d	Testing		Training		Validation	
			ξ	η	ξ	η	ξ	η
2	3	21	70.37%	40.32%	80.65%	40.08%	86.89%	43.68%
	5	45	72.22%	40.15%	80.66%	43.81%	82.43%	42.64%
	8	96	72.07%	40.71%	81.55%	44.02%	79.00%	43.13%
4	3	105	73.56%	44.98%	82.54%	41.94%	86.30%	45.45%
	5	665	**74.92%**	**53.64%**	**83.45%**	**40.66%**	**90.07%**	**50.45%**

The clinical requirements of classification suggest that the value of sensitivity should be large then 70% and the value of specificity should be large then 50%. From the result reported in Table 6.3, we can see that the FIS model trained with the proposed intelligent optimiser can achieve a specificity > 53% and sensitivity > 74%. The testing result can meet the clinical requirements successfully. From the result reported in Table 6.2, we can see that the FIS model trained with the proposed DWM-DE can achieve a specificity > 51% and sensitivity >73%. The testing result also satisfies the clinical requirements. The resulting specificity and sensitivity increase when the number of inputs to the FIS increases. It is reasonable as the FIS can capture more body information to perform the classification. Besides, the resulting specificity and sensitivity increase when the number of membership functions involved in the FIS increases. It is also reasonable as the FIS can have more freedom to model the relationship between the body signals and the presence of hypoglycaemia. The drawback of increasing the number of inputs

and the number of membership functions is that the demand of the computational power for the training process is increased. Yet, the proposed DWM-DE and intelligent optimiser can be trained process successfully and are able to offer better solution quality and reliability.

Table 6.4. Best Testing Results for Hypoglycaemic Detection from Different Approaches.

Method	Sensitivity	Specificity
The *Intelligent Optimiser* with FIS - 4 inputs	76.92%	56.14%
The *Intelligent Optimiser* with FIS – 2 inputs	75.12%	45.32%
The *DWM-DE* with FIS - 4 inputs	75.92%	55.14%
The *DWM-DE* with FIS – 2 inputs	74.92%	47.12%
FIS-4-w/o-v	75.00%	51.64%
FIS-2- w/o-v	73.21%	52.58%
LR-4	51.78%	51.64%
LR-2	50.00%	51.17%
FFNN-2	64.26%	52.50%
MR-2	62.31%	53.10%

IV CONCLUSION

In this chapter, the proposed intelligent optimiser and the proposed DWM-DE algorithm are applied to the hypoglycaemic detection problem. To tackle this problem, a Fuzzy Inference System (FIS) is used as a detector to recognise the presence of hypoglycaemia. The FIS is developed to measure some physiological signals continuously from the human body. It captures the relationship between the presence of hypoglycaemia episodes and the physiological signals of corrected QT interval of the electrocardiogram (ECG) signal and heart rate. The proposed DWM-DE and intelligent optimiser are employed to optimise the FIS parameter values that formulate the fuzzy rules and fuzzy membership functions. Data of 15 children with TIDM are studied and used in the training and testing process for the proposed FIS. Experiment results show that the two proposed optimisation methods could offer good performance on training the FIS. The resulting FIS can offer good performance on doing hypoglycaemia detection. Moreover, the proposed DWM-DE and intelligent optimiser trained the FIS model well for doing classification.

Chapter 7

CONCLUSION

I ACHIEVEMENT

In this thesis, we have proposed two Evolutionary Computation (EC) techniques to solve optimisation problems. They are the Differential Evolution with Double Wavelet Mutations (DWM-DE) algorithm and the intelligent optimiser. The two techniques are developed based on the standard Differential Evolution (DE) algorithm to achieve better searching performance. The proposed DWM-DE algorithm employs two additional stages of wavelet operations. Taking advantage of the wavelet function's properties, the proposed DWM-DE algorithm can offer better performance in terms of solution reliability, solution quality and convergence rate. The proposed intelligent optimiser integrates two DE engines into one single system. The two DE engines share the population information with each other to achieve better searching performance. An internal fuzzy controller is embedded to adjust the internal parameters of the DE engines adaptively. Thanks to the population information sharing and the adaptive control of the internal parameters, the proposed intelligent optimiser can offer much better performance in term of solution quality, solution reliability, and convergence rate. The DWM-DE algorithm and the intelligent optimiser have been applied to the economic load dispatch with valve-point loading (ELD-VPL) problem and the hypoglycaemia detection problem to evaluate their

performance. Experiment results have shown that the two proposed methods can achieve good performance in these two industrial applications.

The design detail of the DWM-DE algorithm is discussed in Chapter 3. To realise the evolution, two stages of wavelet operation are embedded in the standard DE algorithm. By introducing the double wavelet mutations in DE, the searching process is enhanced by offering an effective balance between the exploration and exploitation of the solution space for better solution reliability and quality. In the DE mutation operation, a wavelet function is employed to control the mutation factor F. In the DE crossover operation, a wavelet-based second mutation mechanism is proposed to modify the trial vectors within the population. A suite of 29 benchmark test functions is employed to test the performance of the proposed DWM-DE. The experiment results show that the proposed DWM-DE is a useful tool for solving optimisation problems, and it offers better results in terms of solution reliability, solution quality and convergence rate. The experiment results reflect that DWM-DE is particularly suitable for complex problems with a high dimension ($\geq$20).

The design detail of the proposed intelligent optimiser is discussed in Chapter 4. It incorporates two identical wavelet-mutated Differential Evolution (WM-DE) engines to construct a reliable optimiser. The two engines operate in parallel with the same fitness function. A fuzzy controller is employed in the intelligent optimiser to control the internal parameters of the optimiser. This implementation framework takes advantage of the parallel structure to enhance the optimisation performance. A suite of 29 benchmark test functions is employed to test the performance of the proposed intelligent optimiser. The experiment results show that the proposed intelligent optimiser a useful tool for solving optimisation problems, and it offers better

results in terms of solution reliability, solution quality and convergence rate. In particular, the experiment results show that the intelligent optimiser could offer much better results when the problem is complex and the problem dimension is high (>30).

Two industrial applications of solving the ELD-VPL problem and the hypoglycaemia detection problem are discussed in Chapter 5 and Chapter 6 respectively. The proposed DWM-DE and intelligent optimiser are applied to search for the minimum operating cost for the ELD-VPL problem. Two different requirements of the ELD-VPL problem have been tested. It is observed that the two proposed methods give satisfactory optimal costs when compared with other techniques in the literature. Moreover, the experiment results show that the two proposed methods could offer better performance than the conventional methods in terms of solution quality, convergence speed and solution reliability. Thanks to the wavelet operations, the DWE-DE algorithm searches the solution space more effectively. Thanks to the fuzzy controller and the T-Test analysis in the proposed intelligent optimiser, the population information can be captured to change the parameter values of the optimiser adaptively in order to obtain better searching performance.

For the hypoglycaemia detection problem, a fuzzy inference system (FIS) is employed as a classifier to classify the presence of hypoglycaemic episodes for Type 1 diabetes mellitus (TIDM) patients. A detector is developed to measure some physiological signals continuously from human body. The FIS captures the relationship between the presence of hypoglycaemic episodes and the physiological signals of corrected QT interval of the electrocardiogram (ECG) signal and heart rate. The proposed DWM-DE and intelligent optimiser are employed to optimise the FIS parameter values that formulate the fuzzy rules and fuzzy membership

functions. The data of 15 children with TIDM are studied and used in the training and testing process for the proposed FIS. The experiment results show that the two proposed optimisation methods could offer good performance on training the FIS. The resulting FIS can offer good performance on doing hypoglycaemia detection.

II FUTURE WORK

In this thesis, we have proposed methods applying to the standard DE's crossover and mutation operations in order to enhance the searching performance for better solution quality, solution reliably and convergence rate. In the future, different operation schemes for the DE crossover and mutation operation can be studied to improve the searching performance further. The major objectives of the operation scheme are to control the searching process of the population for a balance between the exploration and exploitation of the solution space. In the early stage of searching, we want more exploration while more exploitation is desired at the later stage. For the intelligent optimiser, we have proposed to use the T-Test algorithm to analyse the population difference between the two engines. Different kinds of statistical algorithms can also be applied to analyse the population difference, for example, the F-Test and the Z-Test. Moreover, different kinds of fuzzy membership functions and fuzzy rules can be investigated to enhance the performance of the intelligent optimiser further. For the industrial application of the hypoglycaemia detection problem, different kinds of modelling technique can be employed to realise the classifier. For example type-2 fuzzy inference system, support vector machines (SVM) and neural networks (NN) can be studied.

REFERENCES

[Adler 77]

R.B. Adler and R. Fischl, "Security constrained economic dispatch with participation factors based on worst case bus load variations," *IEEE Transactions on Power Apparatus and Systems*, vol. 96, no. 2, pp. 347-356, Mar. 1977.

[Ahmed 05]

A.A.E. Ahmed, L.T. Germano, and Z.C. Antonio, "A hybrid particle swarm optimization applied to loss power minimization," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 859-866, May 2005.

[Ahn 01]

C.W. Ahn, R.S. Ramarkrishna, C. G. Kang, and I.C. Choi, "Shortest path routing algorithm using hopfield neural network," *Electronics Letters*, Vol. 37 no. 19, pp. 1176-1178, Aug. 2001.

[Ahn 02]

C.W. Ahn and R.S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 566-579, Jan. 2002.

[Ali 05]

M.M. Ali, C. Khompatraporn and Z.B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," *J. Global Optim.*, vol. 31, pp. 635-672, Apr. 2005.

[Altman 94]

D.G. Altman and J.M. Bland, "Statistics notes: diagnostic tests 1: sensitivity and specificity," *Br. Med. J.*, vol. 308, pp. 1552, Jun. 1994.

[Amir 07]

O. Amir, D. Weinstein, S. Ziberman, M. Less, D. Perl-Treves, H. Primack, A. Weinstein, E. Gabis, B. Fikhte, and A. Karasik, "Continuous noninvasive glucose monitoring technology based on 'occlusion spectroscopy," *J. Diabetes Sci. Technol.*, vol. 1, no. 4, pp. 463-469, Jun. 2007.

[Angeline 98]

P. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *Proc. 7th Conf. Evol. Program.*, 1998, pp.601-610.

[Ao 09]

Y. Ao, and H. Chi, "Experimental study on differential evolution strategies," in *Proc. Global Congress on Intelligent Systems*, May 2009, vol. 2, pp. 19-24.

[Ao 12]

Y. Ao and H. Chi, "Differential evolution using opposite point for global numerical optimization," *Journal of Intelligent Learning Systems and Applications*, vol. 4, no.1, pp. 11-19, Feb. 2012.

[Avriel 03]

M. Avriel, "Nonlinear Programming: Analysis and Methods," *Dover Publishing*, 2003.

[Babu 01]

B. Babu, and R. Angira, "Optimization of non-linear functions using evolutionary computation," in *Proc. 12th ISME International Conference on Mechanical Engineering*, India, 2001, pp. 153-157.

[Bonabeau 99]

E. Bonabeau, M. Dorigo and G. Theraulaz, "Swarm intelligence: from natural to artificial systems," *Oxford Univ. Press*, New York, 1999.

[Bosman 03]

P.A.N. Bosman, "Design and application of iterated density-estimation evolutionary algorithms," Doctoral dissertation, *Utrecht University, TB Utrecht, The Netherlands*, 2003.

[Bratton 07]

D. Bratton and J. Kennedy "Defining a standard for particle swarm optimization," in *Proc. IEEE Swarm Intel. Symp.*, Apr. 2007, pp.120 -127.

[Brest 06]

J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646-657, Dec. 2006.

[Bui 82]

R.T. Bui and S. Ghaderpanah, "Real power rescheduling and security assessment," *IEEE Trans. Power Apparatus Syst.*, Vol. PAS-101, no. 8, pp. 2906-2915, Aug. 1982.

[Caduff 09]

A. Caduff, M.S. Talary, M. Mueller, F. Dewarrant, J. Klisic, M. Donath,L. Heinemann, and W.A. Stahel, "Non-invasive glucose monitoring in patient with type 1 diabetes: A multisensory system combining sensors for dielectric and optical characterization of skin," *Biosens. Bioelectron.*, vol. 24, pp. 2778-2784, 2009.

[Chakrakraborty 08]

U.K. Chakraborty, "Advances in differential evolution," Springer, *Heidelberg*, 2008.

[Chan 11]

K.Y. Chan, C.K. Kwong, T.S. Dillon and Y.C. Tsim, "Reducing overfitting in manufacturing process modeling using a backward elimination based genetic programming," *Applied Soft Computing.*, vol. 11, no. 2, pp. 1648-1656, Mar. 2011.

[Chan 11]

K.Y. Chan, , T.S. Dillon and C.K. Kwong, "Modeling of a liquid epoxy molding process using a particle swarm optimization-based fuzzy regression approach," *IEEE Trans. Industrial Informatics.*, vol. 7, no. 1, pp. 148-158, Feb. 2011.

[Chan 12]

K. Chan, S. Ling, T. Dillon, and H. Nguyen, "Diagnosis of hypoglycemic episodes using a neural network based rule discovery system", *Expert Systems with Applications*, vol. 38, no. 8, pp. 9799-9808, Aug. 2012.

[Chen 01]

C.L. Chen and N. Chen, "Direct search method for solving economic dispatch problem considering transmission capacity constraints," *IEEE Trans. Power Syst.*, Vol. 16, no. 4, pp. 764-769, Nov. 2001.

[Chen 07]

Y. Chen, W. Peng and M. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 6, pp.1460 -1470, Dec. 2007.

[Chen 95]

P.H. Chen and H.C. Chang, "Large-scale economic dispatch by genetic algorithms," *IEEE Trans. Power Syst.*, vol. 10, no. 1, pp. 117-124, Feb. 1995.

[Chiong 12]

R. Chiong, T. Weise, and Z. Michalewicz, "Variants of evolutionary algorithms for real-world applications," *Springer*, 2012.

[Cho 08]

B.H. Cho, H.Yu, K.W. Kim, T.H. Kim, I.Y. Kim, and S.I. Kim, "Application of irregular and unbalanced data to predict diabetic nephropathy using visualization and feature selection methods," *Artif. Intel. Med.*, vol. 42, no. 1, pp. 37-53, Jan. 2008.

[Chu 08]

A. Chu, H. Ahn, B. Halwan, B. Kalmin, E. L. V. Artifon, A. Barkun, M. G. Lagoudakis, and A. Kumar, "A decision support system to facilitate management of patients with acute gastrointestinal bleeding," *Artif. Intel. Med.*, vol. 42, no. 3, pp. 247-259, Mar. 2008.

[Clerc 02]

M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, Feb. 2002.

[Coelho 06]

L.S. Coelho and V.C. Mariani, "Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect," *IEEE Trans. Power systems*, vol. 21, no. 2, pp. 989-995, May. 2006.

[Das 11]

S. Das and P. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4-31, Feb. 2011.

[Daubechies 92]

I. Daubechies, "Ten lectures on wavelets," *Philadelphia, PA: Society for Industrial and Applied Mathematics*, 1992.

[DCCT 95]

DCCT Research Group, "Adverse events and their association with treatment regimens in the diabetes control and complications trial," *Diabetes Care*, vol. 18, pp. 1415-1427, Nov. 1995.

[Directnet 06]

Directnet Study Group, "Evaluation of factors affecting CGMS calibration," *Diabetes Technol. Ther.*, vol. 8, pp. 318-325, Jun. 2006.

[Dorigo 97]

M. Dorigo, and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, Apr. 1997.

[Dorigo 99]

M. Dorigo, and G.Di Caro, "The ant colony optimization metaheuristic," In D. Corne, M, Dorigo, F. Glover (Eds.), new Ideals in optimization, *McGraw-Hill*, 1999.

[Eberhart 00]

R.C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. Congress on Evolutionary Computing*, Jul. 2000, vol. 1, pp.84-88.

[Eberhart 95]

R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th International Symposium on Micro Machine and Human Science*, Nagoya, Oct. 1995, pp.39-43.

[Eberhart 98]

R.C. Eberhart, and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in Evolutionary Programming VII. New York: Springer-Verlag, 1998, vol. 1447, *Lecture Notes in Computer Science*, pp. 611-616.

[Eftekhari 02]

M.M. Eftekhari and L.D. Marjanovic. "Application of fuzzy control in naturally ventilated buildings for summer conditions," *Energy and Buildings*, vol. 35, no. 7, pp.645-655, Aug. 2003.

[El-Sharkawy 96]

M. El-Sharkawy and D. Nieebur, "Artificial neural networks with application to power systems," *IEEE Power Engineering Society, A Tutorial Course*, 1996.

[Esmin 05]

A. Esmin, G. Lambert-Torres, and A. Zambroni "A hybrid particle swarm optimization applied to loss power minimization," *IEEE Trans. Power Syst*., vol. 20, no. 2, pp.859 -866, May 2005.

[Fan 03]

H.Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *J. Global Optim.*, vol. 27, no. 1, pp. 105-129, Sep. 2003.

[Farag 95]

A, Farag, S. Al-Baiyat, and T.C. Cheng, "Economic load dispatch multi-objective optimization procedures using linear programming techniques," *IEEE Trans. Power Syst.*, Vol. 10, pp. 731-738, May 1995.

[Fogel 94]

L. Fogel, "Evolutionary programming in perspective: The top-down view," *Computational Intelligence: Imitating Life. Piscataway, NJ: IEEE Press*, 1994.

[Freedman 05]

D.A. Freedman, "Statistical Models: Theory and Practice," *Cambridge, U.K.: Cambridge Univ. Press*, 2005.

[Gaing 04]

Z. Gaing, "A Particle Swarm Optimization Approach for Optimum Design of PID Controller in AVR System," *IEEE Transactions on Energy Cconversion*, vol. 19, no. 2, pp. 384-391, Jun. 2004.

[Goldbreg 89]

D.E. Goldbreg, "Genetic algorithms in search, optimization, and machine learning. Reading," *MA: Addison-Wesley*, 1989.

[Grabot 97]

B. Grabot, L. Geneste, and A. Dupeux, "Tuning of fuzzy rules for multi-objective scheduling, Fuzzy Information Engineering: A Guided Tour of Applications," pp. 695-703, *Wiley*, 1997.

[Gudise 04]

V. Gudise and G. Venayagamoorthy, "FPGA placement and routing using particle swarm optimization," in *Proc IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI System Design*, Piscataway, 2004, pp. 307-308.

[Hsieh 09]

S. Hsieh, T. Sun, C. Liu, and S. Tsai, "Efficient population utilization strategy for particle swarm optimizer," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 2, pp.444-456, Apr. 2009.

[Jens 10]

K. Jens, D.R. Graeme, and K. Stefan, "Swarm intelligence in animals and humans," *Trends in Ecology and Evolution*, vol. 25, no.1, pp.28-34, Jan. 2010.

[Juang 04]

F. Juang, "A hybrid genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Systems, Man and Cybernetics: Part B: Cybernetics*, vol. 34, no. 2, pp. 997-1006, Apr. 2004.

[Kennedy 01]

J. Kennedy and R. Eberhart, "Swarm intelligence," *Morgan Kaufmann Publishers*, 2001.

[Kennedy 02]

J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. Congr. Evol. Comput.*, May 2002, pp.1671 -1676.

[Kennedy 95]

J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, 1995, vol. 4, pp.1942-1948.

[Kennedy 97]

J. Kennedy and R.C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. Int. Conf. Systems, Man, Cybernetics*, 2007, pp. 4104-4109.

[Kennedy 99]

J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance", in *Proc. IEEE Int. CEC*, 1999, vol. 3, pp.1931 -1938.

[Klir 88]

G.J. Klir and T.A. Folger, "Fuzzy Sets, Uncertainty, and Information," *Prentice-Hall*, 1988.

[Labbi 05]

Y. Labbi and D.B. Attous, "A hybrid ga-ps method to solve the economic load dispatch problem," *Journal of Theoritical and Applied Information Technology*, vol. 1, no. 1, pp. 61-65, Oct. 2005.

[Li 12]

C. Li, S. Yang, and T. Nguyen, "A self learning particle swarm optimizer for global optimization problems", *IEEE Transactions on Systems, Man and Cybernetics,* vol. 42, no. 3, pp. 627-646, Jun. 2012.

[Liang 01]

Q. Liang and J.M. Mendel, "MPEG VBR video traffic modeling and classification using fuzzy technique," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 1, pp. 183-193, Feb. 2001.

[Liang 06]

J.J. Liang , A.K. Qin , P.N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp.281 -295, Jun. 2006.

[Ling 08]

S.H. Ling, H.H.C. Iu, K.Y. Chan, H.K. Lam, C.W. Yeung, and F.H.F. Leung, "Hybrid particle swarm optimization with wavelet mutation and its industrial applications," *IEEE Trans. Syst., Man and Cybern., Part B: Cybernetics,* vol. 38, no. 3, pp. 743-763, Jun. 2008.

[Ling 11]

S.H. Ling and H.T. Nguyen, "Genetic-algorithm-based multiple regression with fuzzy inference system for detection of nocturnal hypoglycemic episodes," *IEEE Trans On Information Technology In Biomedicine.*, vol. 15, no. 2, pp. 308-315, Feb. 2011.

[Ling 12]

S.H. Ling, and H.T. Nguyen, "Natural occurrence of nocturnal hypoglycemia detection using hybrid particle swarm optimized fuzzy reasoning model," *Artificial Intelligence in Medicine*, vol. 55, no. 3, pp. 177-184, Jul. 2012.

[Liu 06]

H. Liu, P. Li, and Y. Wen, "Parallel ant colony optimization algorithm," in *Proc. Intelligent Control and Automation, WCICA*, vol. 1, 2006, pp. 3222-3226.

[Liu 07]

D. Liu, K. C. Tan , C. K. Goh and W. K. Ho "A multiobjective memetic algorithm based on particle swarm optimization", *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 1, pp.42 -50, Feb. 2007.

[Maia 07]

F.F.R. Maia and L.R. Araujo, "Efficacy of continuous glucose monitoring system (CGMS) to detect postprandial hyperglycemia and unrecognized hypoglycemia in type1 diabetic patients," *Diabetes Res. Clin. Pract.*, vol. 75, pp. 30-34, Jan. 2007.

[Maield 94]

T. Maield and G. Sheble, "Short-term load forecasting by a neural network and a reined genetic algorithm," *EPSR*. vol. 31, no. 2, pp. 147-152, Dec. 1994.

[Mendes 04]

R. Mendes, "Population topologies and their influence in particle swarm performance," Doctoral dissertation, *Universidade do Minho*, 2004.

[Michalewicz 94]

Z. Michalewicz, "Genetic Algorithm + Data Structures = Evolution Programs," 2nd ed., *New York: Springer-Verlag*, 1994.

[Miller 10]

P. Miller, "The Smart Swarm: How understanding flocks, schools, and colonies can make us better at communicating, decision making, and getting things done," *New York: Avery*, 2010.

[Mo 07]

N. Mo, Z.Y. Zou, K.W. Chan, and T.Y.G. Pong, "Transient stability constrained optimal power flow using particle swarm optimization," *IEE Generation, Transmission and Distribution*, vol. 1, no. 3, pp.476-483, May 2007.

[Naka 03]

S. Naka, T. Genji, T. Yura, and Y. Fukuyama, "A hybrid particle swarm optimization for distribution state estimation," *IEEE Trans. Power Syst.*

[Neubauer 97]

A. Neubauer, "A theoretical analysis of the non-uniform mutation operator for the modified genetic algorithm," in Proc *IEEE Int. Conf. Evol. Comput*., Indianapolis, IN, Aug. 1997, pp. 93-96.

[Nocedal 06]

J Nocedal and S.J. Wright, "Numerical optimization," *New York: Springer*, 2006.

[Noel 04]

M.M. Noel and T.C. Jannett, "Simulation of a new hybrid particle swarm optimization algorithm," in *Proc 36th South eastern Symposium on System Theory*, 2004, pp.150-153.

[Nuryani 12]

N. Nuryani, S. Ling, and H. Nguyen, "Electrocardiographic signals and swarm-based support vector machine for hypoglycemia detection," *Annals Of Biomedical Engineering*, vol. 40, no. 4, pp. 934-945, Apr. 2012.

[Pancholi 04]

R.K. Pancholi, and K.S. Swarup, "Particle swarm optimization for security constrained economic dispatch," in *Proc. International Conference on Intelligent Sensing and Information Processing*, Chennai, India, 2004, pp.7-12.

[Park 05]

J.B. Park, K.S. Lee, and K.Y. Lee, "A particle swarm optimization for economic dispatch with non-smooth cost functions," *IEEE Trans. Power systems*, vol. 20, pp. 34-49, Jan. 2005.

[Paterini 04]

S. Paterlini and T. Krink, "High performance clustering with differential evolution," in *Proc. IEEE Congress on Evolutionary Computation*, vol. 2, 2004, pp. 2004-2011.

[Pelikan 02]

M. Pelikan, "Bayesian optimization algorithm: from single level to hierarchy," Doctoral dissertation, *University of Illinois at Urbana-Champaign, Urbana*, 2002.

[Pickip 00]

J.C. Pickup, "Sensitivity glucose sensing in diabetes," *Lancet*., vol. 355, pp. 426-427, 2000.

[Poli 07]

R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intell.*, vol. 1, no. 1, pp.33-57, Jun. 2007.

[Qing 06]

A. Qing, "Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems," *IEEE Trans. Geoscience and Remote Sensing*, vol. 44, issue 1, pp. 116-125, Jan. 2006.

[Rahnamayan 08]

S. Rahnamayan, H.R. Tizhoosh, and M.M.A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp 64-79, Feb. 2008.

[Reeves 94]

C.R. Reeves, "Genetic algorithms and neighbourhood search," in *Proc Evolutionary Computing: AISB Workshop*, 1994, pp. 115-130.

[Rifaie 12]

M. Rifaie, J.M. Bishop, and S. Caines, "Creativity and autonomy in swarm intelligence systems," *Cognitive Computing*, vol. 4, no. 3, pp 320-331, Feb. 2012.

[Sabhin 05]

E. Sabhin, "Swarm Robotics: From Sources of Inspiration to Domains of Application", *ser. Lecture Notes in Computer Science, Berlin/Heidelberg: Springer*, vol. 3342, pp. 10-20, 2005.

[Seber 08]

G.A.F. Seber, "Linear Regression Analysis," *New York: Wiley*, 2003.

[Selvakumar 07]

A.I. Selvakumar and K. Thanushkodi, "A New Particle Swarm Optimization Solution to Non convex Economic Dispatch Problems", *IEEE Trans. Power systems*, vol. 22, no.1, pp. 42-50, Jan. 2007.

[Srinivas 94]

M. Srinivas, and L. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on System, Man and Cybernetics*, vol.24, no.4, pp.656-667, Apr, 1994.

[Storn 97]

R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, Dec. 1997.

[Swagatam 2008]

D. Swagatam, A. Ajith, and K. Amit, "Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives," *Springer-Verlag Berlin Heidelberg* 2008.

[Vaessens 92]

R.J.M. Vaessens, E.H.L. Aarts, and J.K. Lenstra, "A local search template," in *Proc. Parallel Problem Solving Nature*, 1992, vol. 2, pp. 65-74.

[Van Sickel 07]

J.H. van Sickel, K.Y. Lee, and J.S. Heo, "Differential evolution and its applications to power plant control," in *Proc. Intelligent Systems Applications to Power Systems*, Nov. 2007, pp.1 - 6.

[Vesterstroem 04]

J. Vesterstroem and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. Congr. Evol. Comput.*, 2004, vol. 2, pp. 1980-1987.

[Vicsek 08]

T. Vicsek, “Universal patterns of collective motion from minimal models of flocking,” in *Proc. IEEE conference on Self-Adaptive and Self-Organizing Systems*, 2008, pp. 3-11.

[Victoire 04]

T.A.A. Victoire and A.E. Jeyakumar, “Hybrid pso-sqp for economic dispatch with valve-point effect,” *Electrical Power Systems Research*, vol. 71, pp. 51-59, Sep. 2004.

[Wang 06]

S. Wang and W. Min, “A new detection algorithm (NDA) based on fuzzy cellular neural networks for white blood cell detection,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 10, no. 1, pp. 5-10, Jan. 2006.

[Wang 2008]

C. Wang, Z. Li, and S. Zhu, “Minimizing total tardiness on parallel machines based on Genetic Algorithm,” in *Proc. Control and Decision Conference,* 2008, pp.165 – 169.

[Weinstein 07]

R. L. Weinstein, “Accuracy of the freestyle navigator CGMS: Comparison with frequent laboratory measurements,” *Diabetes Care*, vol. 30, pp. 1125-1130, 2007.

[Whitley 91]

D. Whitley, K. Mathias, and P. Fitzhorn, L. Booker and R. Belew, “Delta coding: An iterative search strategy for genetic algorithms,” in *Proc. 4th Int. Conf. Genetic Algorithms*, 1991, pp.77-84.

[Wu 96]

K.C. Wu, “Fuzzy interval control of mobile robots,” *Computers and Electrical Engineering*, vol. 22, no. 3, pp. 211-229, May 1996.

[Yalcinoz 98]

T. Yalcinoz and M.J. Short, "Neural networks approach for solving economic dispatch problem with transmission capacity constraints," *IEEE Trans. Power Syst.*, vol. 13, pp. 307-313, May 1998.

[Yale 04]

J.F. Yale, "Nocturnal hypoglycemia in patients with insulin-treated diabetes," *Diabetes Res. Clin. Pract.*, vol. 65, pp. S41-S46, Sep. 2004.

[Yao 99]

X. Yao and Y. Liu, "Evolutionary programming made faster," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, July 1999.

[Yoshida 00]

H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Trans. Power Syst.*, vol. 15, no. 4, pp.1232 -1239, Nov. 2000.

[Youssef 00]

H.K. Youssef, and K.M., El-Naggar, "Genetic based algorithm for security constrained power system economic dispatch," *Elec. Power Syst. Res.*, Vol. 53, issue 1, pp. 47-51, Jan. 2000.

[Yu 07]

B. Yu, X. Yuan, and J. Wang, "Short-term hydro-thermal scheduling using particle swarm optimization method," *Energy Convers. Manage.*, vol. 48, no. 7, pp.1902 -1908, Jul. 2007.

[Yuan 2011]

Y. Yuan, Z. Kailong, and Z. Xingshe, "Application of a self-organizing fuzzy neural network controller with group-based genetic algorithm to greenhouse," in *Proc. 7th International Conference Natural Computation*, Oct. 2006, pp. 529 - 534.

[Zahan 99]

S. Zahan, C. Micheal, and S. Nikolakeas, "Fuzzy and neuro-fuzzy systems in medicine," *CRC*, 1999.

[Zhang 05]

Q. Zhang, J. Sun, and E. Tsang, "An evolutionary algorithm with guided mutation for the maximum clique problem," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 192-200, Apr. 2005.

[Zhang 07]

J. Zhang, H. Chung, and W. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," *IEEE Transactions on Evolutionary Computation, vol*. 11, no. 3, pp. 326-335, Jun. 2007.

[Zhao 05]

B. Zhao, C.X. Guo, and Y.J. Cao, "A multiagent-based particle swarm optimization approach for optimal reactive power dispatch," *IEEE Trans. Power Systems.*, vol. 20, no. 2, pp. 1070-1078, May 2005.